Amazon Flexible Payments Service
Advanced Quick Start Guide
API Version 2010-08-28

Table of Contents

Welcome

This guide describes the concepts for and gives instructions to set up multiple or recurring payments between a buyer and a seller for e-commerce, digital content, donations, or services.

Amazon Flexible Payments Service (FPS) is a web service that enables developers to accept payments on their website. The payments can be for selling goods or services, raise donations, execute recurring payments, and send payments.

How Do I...?

How do I?	Relevant Sections
Decide whether Amazon FPS is right for my needs	Amazon FPS detail page
Get started with Amazon FPS quickly	Amazon Flexible Payments Service Getting Started Guide
Learn the key concepts for this service	Introduction to Amazon FPS Advanced Quick Start
Enable merchants to register on your web site so they can receive payment for their items sold through your web site	Recipient Registration
Enable buyers to authorize payments	Getting Authorization
Handle refunds and cancellations	Order Cancellations
Find reference information on the actions provided by this quick start	Amazon FPS API Reference and the Co-Branded Service API Reference
Find examples for creating signatures and making a pay request	Code Samples

Introduction to Amazon FPS Advanced Quick Start

This introduction to Amazon FPS Advanced Quick Start provides a detailed summary of this web service. After reading this section, you should have a good idea of what it offers and how it can fit in with your business.

Overview

This overview describes the business model and major features of Amazon FPS Advanced Quick Start.

Business Model

Amazon FPS Advanced Quick Start enables buyers to authorize a payment token that can be used more than one time. For more information about payment tokens, see “Payment Token Types.”

These payment tokens, which are set up with a certain amount of money authorized, might be recurring payment tokens, which are charged a fixed price at regular intervals. Or they could be multi-use tokens, which are charged according to the token's constraints. For example, the token could be charged a certain number of times per day, or for a certain number of users, or until the authorization runs out of money.

These token types are highly configurable. For a recurring payment token, you can configure how often and how much is paid, and when the token expires. For a multiple payment token, you can configure its expiration, the merchants it can be used to pay, the maximum and minimum amount it can be charged at any one time, how many times it can be charged, the lifetime maximum amount it can be charged, and so on.

Features

Amazon FPS Advanced Quick Start provides the following major features:

Payment authorization—Buyers must authorize payments for you to be able to charge them. You can get this authorization by redirecting them to the Amazon Payments website. Buyers log in to Amazon Payments and agree to make the payment using a specified payment method.

Payments—Transfer money from the buyer's account to the seller's.

Recurring-use payment tokens—A buyer authorizes a recurring payment token once, and thereafter it is charged a specific amount at regular intervals until the token expires.

Multi-use payment tokens—A buyer authorizes a multi-use payment token once but it can be used repeatedly according to its constraints, such as the maximum number of times it can be used within a given period of time.

Seamless mobile payment support—Amazon FPS automatically supports payments within mobile web browsers.

Notifications—Get notified automatically when transactions succeed or fail.

Refunds—Refund the money from a successfully completed transaction..

Cancellation of payment tokens—Cancel any of your payment tokens at any time.

Additional FPS Quick Start Guides

For different functionality, such as multi-use payment tokens, go to one of the other Amazon FPS Quick Starts.

Amazon FPS has four Quick Start implementations, each providing a different slice of Amazon FPS functionality:

How do I?	Relevant Sections
Basic	Facilitates one-time payment between a buyer and a developer who is also the merchant for e-commerce, digital content, donations, services.
Marketplace	Facilitates one-time payment between buyer and merchant where you are a third party developer, a caller, who hosts the merchant's products and order pipeline. With a unique three-party transaction model, payments can be processed in which you are neither the buyer nor the seller. You can charge a fee for such transactions.
Advanced	Facilitates multiple or recurring payments.
Account Management	Accesses buyer and developer account activity programmatically. Alternatively, view account activity and balances can be viewed on the the Amazon Payments website.

You can use these parts separately or in combination. They share a common WSDL and schema.

Key Concepts

This section describes the concepts and terminology you need to understand to use Amazon FPS Advanced Quick Start effectively.

Amazon FPS Advanced Quick Start

Amazon FPS Advanced Quick Start enables you to create multi-use payment tokens with only a single authorization, and to create innovative payment solutions that allow payments between two applications.

Amazon Flexible Payments Service

Amazon FPS is the first payments service designed from the ground up specifically for developers. This set of web service APIs differs from other Amazon Payments products, such as Amazon Simple Pay and Checkout by Amazon, in that it allows the development of highly customized payment solutions for a variety of businesses.

Amazon FPS is built on top of Amazon's reliable and scalable payments infrastructure and provides developers with a convenient way to charge the tens of millions of Amazon customers. Amazon customers can pay using the same login credentials, shipping address and payment information they already have on file with Amazon.

For buyers, the advantage of using Amazon FPS payment instruments in online purchases includes the following:

Convenience—Consumers can use their Amazon.com account to complete payments on a website without having to re-enter their shipping address or payment information.

Trusted payment experience—The secure and trusted payment experience consumers enjoy on Amazon.com is available for your website.

Purchase protection for buyers—Consumers can feel more confident purchasing, knowing that they have the same protection under the Amazon A-to-z Guarantee that they have when they shop on Amazon.com.

For sellers, the advantage of using Amazon FPS includes the following:

Flexibility—Amazon FPS offers immense flexibility by allowing you to define terms and conditions specific to each transaction. It also gives you control over when the payment transaction is executed.

Access to Amazon customers—Amazon FPS enables tens of millions of existing Amazon customers to transact online, simply using the same accounts and payment methods that they use for purchases on Amazon.com.

Increased customer base—Amazon's trusted payment experience, A-to-z Guarantee, and the ease with which tens of millions of Amazon customers can pay on a website will help increase the total number of Amazon customers.

Lower cost with Amazon's proven fraud detection—Amazon FPS leverages Amazon's proven fraud detection capabilities, chargeback controls, and risk management processes to reduce bad debt.

Reliable and secure payments platform—Amazon has spent over a decade developing, testing, and operating a reliable, scalable and secure payments infrastructure to support millions of daily transactions. Amazon FPS exposes this robust infrastructure to you and your customers.

Multi-Use Payment Tokens

The multi-use payment token is a usage-based payment instrument that you can constrain in a few ways or in many ways. For example, you can constrain the allowed charge amount per time period, (such as a day, a week, a month, or the token lifetime).You can also constrain the token to pay at least a minimum amount per transaction, or to be used only a specific number of times in any time period. Or, you can use multi-use payment tokens to pay more than one recipient.

You can use the multi-use payment token for open-ended payments where the total is not known beforehand. You might own a music download company that charges a set fee per month for membership and a fee for each download. Before the subscriber downloads each song, you add the download fee to the monthly membership fee and arrive at the current monthly bill. If the total is within the prescribed boundaries of the multi-use payment token, you can allow the music download and charge the subscriber. If the total exceeds the maximum authorized dollar amount associated with the payment token, you display a message and refuse to download unless the subscriber authorizes additional spending.

You, as the caller, can use this token to charge the sender multiple times to pay one or multiple recipients. You cannot use a multi-use token to charge the sender for an unlimited amount or for unlimited amount of time.

> **Note**
> If you charge your customers at a regular interval for a fixed amount, you should use a recurring token instead. For more information, see "Recurring Payment Tokens."

Usage Restrictions

A multi-use token provides the flexibility of usage based on restrictions or limitations. You can specify two types of restrictions for a multi-use token:

Amount Limit—Limiting the maximum amount This specifies the maximum amount that can be charged using the token within a specified period of time or for the lifetime of the token. For example, the token can be used for a maximum amount of $30 in a month starting from August 10, 2008 or for a maximum amount of $30.

Usage Limit—Limiting the number of uses This specifies the maximum number of times you can charge the token within a specified period of time or for the lifetime of the token. For example, the token can be used a maximum of 10 times or for 10 times per day starting from August 10, 2008.

A maximum of three usage restrictions can be specified for a multi-use token. For example, a combination of usage restrictions mentioned in the preceding list restricts a token to be used 10 times a day for a maximum of $30 in a month.

> **Important**
> Amazon FPS requires you to always set the maximum amount limit.

In addition to the usage restrictions, you can also set one of the following restrictions on the transaction amount. These apply individually to each transaction and not across multiple transactions:

- Maximum transaction amount for each transaction
- Minimum transaction amount for each transaction
- Exact transaction amount for each transaction

A multi-use token can be used to pay multiple recipients in a marketplace application. To enable this, the calling application should specify the list of recipients that will receive payments. Once the list of recipients are specified in the token, it cannot be modified.

Buyer Experience of a Multi-Use Payment Token

The following describes the typical buyer experience of authorizing the multiple payment token and then, later, purchasing items within the limits of that payment token at different times.

Authorizing and Using a Multi-Use Payment Token

1. Your web site takes the buyer through the checkout process.
2. If the buyer chooses Amazon Payments, your **Pay Now** button code sends a co-branded service request that includes the multi-use payment token parameters, and also redirects the buyer to the CBUI web pages that Amazon hosts.
3. The buyer selects a payment instrument and authorizes the purchase.
4. The CBUI redirects the buyer to the URL specified in the returnURL from the co-branded service request. In addition to that URL, the URI contains additional parameters such as the ID for the multi-use token just created (which you need later), the status of the payment authorization, and the identification of the buyer.
5. When the status changes to Success you can use the payment token.
6. At some point, you use the payment token for a purchase without getting additional authorization from the buyer. To facilitate the purchase, you send a Pay request with the multi-use token ID you received earlier.

7. If the buyer makes further purchases you can send additional Pay requests.

Recurring Payment Tokens

Recurring-use payment tokens are a subset of multi-use payment tokens, but with the following restriction types:

- Pre-determined fixed amount (specified at the time of token creation)
- Regular interval

An example would be a subscription for a magazine that charges users $9.99 every month for one year starting on March 10, 2012.

The recurring-use payment token can be charged on a recurring basis, but the buyer only authorizes the payment once. You can customize the token by configuring a variety of parameters. You can set the amount of the recurring payment, the frequency of payment, the starting date of the payments, and the expiration of payments in each payment token.

The advantage of FPS is that you have full control over the charges. For example, you might already be charging a credit card for subscriptions, and you might want to manage an additional payment method, without changing your business processes and control when the payment actually happens.

> **Important**
> You must make a Pay request each time you want to charge a recurring payment token. The constraints associated with the recurring payment token restrict its use only; they do not set up automatic payments.

You might set up recurring payment tokens to pay for such things as membership fees, online newsletters, donations, and loan payments.

Buyer Experience of a Recurring Payment Token

The following describes the sequence of events a buyer goes through while purchasing a music download subscription.

Purchasing a Music Download Subscription

1. The buyer goes to the music subscription web site and clicks a **Subscribe Now** button (or equivalent wording) to purchase the music subscription.
2. The buyer is redirected to the Amazon-hosted Co-branded User Interface (CBUI) pages where he or she logs in, selects the payment instrument, reviews the costs and payment structure, and authorizes the purchase of the subscription.
3. The buyer is redirected to the URL you specify in the Co-branded service request, which is typically a thank you page and an invitation to keep shopping.

4. After the developer has issued the Pay request, the buyer notices that their payment instrument is charged on the start date of the subscription and at the prescribed intervals thereafter.

Recurring Payments

The first point where you must integrate Amazon FPS is at the point where buyer decides to purchase, as described below. For more information, see “Getting Authorization.”

Authorizing and Using a Recurring Payment Token

1. Your web site takes the buyer through the checkout process.
2. If the buyer chooses Amazon Payments, your **Pay Now** button code sends a co-branded service request with the recurring payment token parameters, and redirects the buyer to the CBUI web pages that Amazon hosts.
3. The buyer signs in, selects a payment instrument, and authorizes the purchase.
4. The CBUI redirects the buyer to the URL specified in the returnURL from the co-branded service request.
   In addition to that URL, the URI contains additional parameters such as the token ID for the token just created (which you need in the next step), the status of the payment authorization, and the identification of the buyer.
5. When the status changes to Success, your code constructs and sends Amazon FPS a Pay request to initiate the purchase. The request includes the token ID received in the previous step. You should save the transaction ID that is returned in case of a refund or chargeback.
6. You can send additional Pay requests at regular intervals based on the parameters in the co-branded service request.

Recipient Tokens

Amazon FPS Advanced Quick Start includes the option of using marketplace functionality with recurring-use and multi-use payment tokens. In the marketplace environment, you function as a third-party caller who makes Amazon FPS web service calls to transfer money between a sender and a recipient. To implement marketplace functionality, you must register the recipient (also called the seller or the merchant) on your website. Your registration process creates a recipient token that you later use in a Pay request to transfer money from the buyer to the seller

A recipient token contains the following information:

- Accepted payment method
- Agreement to pay fees (if you don't pay the associated fees)
- Validity of the token
- Authorization for you as the caller to transfer money

Other Integration Points

The preceding sections explain how recurring and multi-use payment tokens fit into the workflow of your web site. The following sections show the other places where Amazon FPS fits into the workflow of your web site.

Cancel Payment

Buyers can cancel recurring and multi-use payment tokens as described below.

Canceling Interaction

1. The buyer clicks the equivalent of a **Cancel** button hosted on your web site.

> **Note**
> A buyer can also cancel a payment at the Amazon Payments web site. If that happens, the merchant is sent an IPN notification of the cancellation.

2. Your **Cancel** button code constructs and sends a CancelToken request to Amazon FPS, which includes the tokenID returned in the co-branded service response.
3. Amazon FPS processes the request and returns an XML notification of the request's success or failure.

Refund Payment

Buyers cannot request a refund through the buyer account page on Amazon Payments. Refunds must be handled through your web site by a request you send to Amazon Payments.

There are two ways you can refund a payment. First, you can visit your account on the Amazon Payments web site to refund the payment. Or, you can issue refunds programmatically using the Amazon FPS Refund action.

Sender, Recipient, and Caller Actions

Participants involved in an Amazon FPS transaction perform one or more of the following actions:

- Send money
  The buyer, known as the sender in an Amazon FPS transaction, makes the payment for purchasing goods or services. The sender can send money using an Amazon Payments Personal account, Amazon Payments Business account, or Amazon FPS developer account.
- Receive money
  The merchant (or seller), also known as the recipient in an Amazon FPS transaction, receives payment for the goods or services sold to the sender. A recipient can receive money using an Amazon Payments Personal account, Amazon Payments Business account, or Amazon FPS developer account.

- Make Amazon web service calls to enable money transfer
  The developer, also known as the caller in an Amazon FPS transaction, can transfer money between a sender and a recipient in a transaction. A caller can also perform the role of a sender or a recipient. A caller must have an Amazon FPS developer account to make web service API calls. For more information about registering for an Amazon FPS account, go to the Amazon Flexible Payments Service Getting Started Guide.

Important
Amazon FPS does not allow a participant to play all three roles in a single transaction.

Request Security

Amazon FPS applications enable payments between buyers and sellers. Web service requests are sent over the Internet using SSL (HTTPS).

HTTPS does not establish the identity of the requester. To establish the identity of the requester, Amazon FPS uses a signature.

A signature is an encrypted value that you generate and include as a parameter value in every request using the signature parameter as in the following example.

```
Signature=K2ryWe7s/0AHI0/PbuAveuUPksTefhmNCzDTold2VYA=
```

With signature version 2, you have the option of using either SHA256 or SHA1 for signature authentication in inbound requests. For outbound notifications, the RSA-SHA1 algorithm is supported.

Important
The previous method for signing (signature version 1) was deprecated on November 3rd, 2009, and as of 10 February, 2011 it is no longer supported. Whenever you sign a request with your access keys, you must now use signature version 2.

Signing is required for all Amazon FPS API requests, and optional but recommended for Co-Branded service requests. If you do not sign a Co-Branded service request, you must manually determine whether the request has been tampered. For detailed information about generating a signature, see “Working with Signatures.”

Co-Branded User Interface (CBUI)

A buyer who is ready to purchase a product clicks a **Buy Now** button powered by Amazon FPS to authorize a payment. The button redirects the buyer from your website to the Co-Branded User Interface (CBUI).

For buyers, the CBUI is a series of web pages where buyers sign in, specify a personal payment instrument, such as a credit card, and then authorize the purchase. Anyone who has purchased something on Amazon.com is familiar with the final approval in the checkout process where you commit to spending your money.

You cannot issue an Amazon FPS Pay request until a buyer has successfully completed the CBUI web pages, and has authorized the purchase.

For the merchant, the CBUI is a series of web pages in which the merchant registers with a caller for a marketplace storefront on the caller's website. Merchant registration is only required in marketplace selling environments. You use the recipient token ID returned from that request to pay merchants in the purchase transaction.

The CBUI enables you to include your company's branding on the CBUI payment authorization web pages. This makes for a better buying experience. Clicking a **Buy Now** button powered by Amazon FPS redirects the buyer away from your website to Amazon's. By including your branding on Amazon's CBUI web pages, buyers don't feel as if they've left the your website to authorize a payment. The CBUI provides continuity between the checkout and payment authorization experience.

For merchant registration, Co-Branding provides a similar, improved customer experience.

Where the CBUI Fits in the Workflow

The following description explains how you redirect buyers to the CBUI web pages when they are ready to purchase the items they selected on your website, the authorization process, and the subsequent Amazon FPS request you make after receiving notification of the authorization.

Authorization and Transaction Process

1. A buyer is ready to checkout, and after entering the shipping address and speed, selects Amazon Payments by clicking the **Pay Now** button. Your website code constructs a Co-Branded service request that identifies the buyer, and sends it when you redirect the buyer to the CBUI web pages.
2. A buyer signs into an Amazon Payments account by entering the proper email and password credentials.
3. The buyer chooses a payment instrument, such as a credit card, bank account, or Amazon Payments balance transfer, to make the purchase.
4. After reviewing the transaction details, the buyer clicks the **Confirm** button to authorize the payment using the specified payment instrument.
5. The Co-Branded service creates a payment token and redirects the buyer to the URL you specify in your Co-Branded service request (in the returnURL parameter). Typically, returnURL contains the URL of a Thank you page in which you invite senders to keep shopping (perhaps by showing them similar items to what they purchased).
6. The URI contains not only the endpoint that you specified in returnURL, but also a reference to the payment token (such as a tokenId), and the status of the authorization.
7. Upon receiving the URI from the Co-Branded service, if the status of the authorization is successful, you must send Amazon FPS a Pay (or Reserve) request to actually transfer

Recipient Registration on Your website

The first step in the workflow is recipient registration on your website. Recipients must register with you so that:

- They can accept your business terms, in particular, the marketplace fee you will charge them
- They can upload their item information to your website
- You can get the RecipienttokenID which you need to facilitate the payment to the recipient

The process describs the process for recipient registration.

Process for Recipient Registration

1. The recipient creates a merchant account on your website. When the recipient clicks the **Register Now** button the code on your website constructs a Co-Branded service request that identifies the recipient, and sends it when you redirect the recipient to the CBUI web pages.
2. On the CBUI the recipient signs into an Amazon Payments account by entering the proper email and password credentials.
3. The recipient confirms the authorization of the payment of fees for your service and the Co-Branded service redirects the recipient back to your website via the returnURL which contains RecipientTokenID.
4. You should store the RecipientTokenID in your database.

Payment Token Types

Every Amazon FPS payment transaction requires a payment token. When someone successfully completes the CBUI web pages, the Co-Branded service creates a payment token. A payment token represents purchase information, including the amount of the purchase, the buyer, and the authorization to use the token as a means of making a purchase. Before you can initiate any Amazon FPS payment transaction, such as a Pay request, you must obtain a token.

There are a number of different kinds of payment tokens and each one has different characteristics. Each Amazon FPS Quick Start implementation provides a set of token types. This Quick Start covers the following:

- **Single-use—**Authorized to make a single purchase of a specified amount where the money is sent from the buyer to you
  Available for: Amazon Flexible Payments Service Basic Quick Start
- **Recurring-use—**Authorized to make payments at regular intervals for such as for subscriptions
  This token can have usage limitations, for example, an expiration date. The payment can be made to you or a third party. In this case, you broker the deal and collect a marketplace fee for doing so. This scenario, in which there are three parties involved,

buyer, merchant, and you, is called a marketplace scenario, and is intended for use with physical products which are shipped to the customer.
This token can be used in a marketplace scenario.
Available for: Amazon Flexible Payments Service Advanced Quick Start

- **Multi-use—**Authorized to be used one or more times within its specified limitations, for example, the total amount it can be used for, how long it can be used, or how little or how much any single payment can be
  This token can be used in a marketplace scenario.
  Available: Amazon Flexible Payments Service Advanced Quick Start
- **Editing—**Authorizes the change of an existing token
  You can edit the multiuse, recurring, and settlement tokens IDs. This enables you to change information in an existing token, for example, the credit card number. If a credit card expires or is replaced, you can use the edit token to modify the recurring token information without having to require the buyer to cancel and re-purchase the item or service.
  Available for: Amazon Flexible Payments Service Advanced Quick Start

Sender and Recipient Token Associations

Buyers, and, in Amazon FPS Quick Start implementations that support the marketplace scenario, merchants, can go through the CBUI to create tokens. Each one does so for a different purpose. The buyer uses a Buy Now button to go through the CBUI to authorize a purchase with a sender token. The recipient uses a Register Now button to authorize the payment of marketplace fees to you for hosting his or her e-commerce store. (For information on marketplace applications, see Amazon Flexible Payments Service Marketplace Quick Start.)

In both cases, it is your website that implements the button that redirects the person to the CBUI.

All of the token types can be associated with a sender, that is, a buyer who is authorizing a purchase. So, there can be a sender single use payment token, or a sender recurring use payment token, and so on. This guide sometimes shortens these names to sender token. The value returned in TokenId from the CBUI is used as the value for SenderTokenId in subsequent Amazon FPS requests.

The token types that can be used in the marketplace scenario can also be associated with a recipient. In this scenario, you host the e-commerce store of a merchant, called a recipient (the person who receives the money).You charge the recipient a fee (called a marketplace fee) for hosting their e-commerce store and brokering the money transactions. On your website, you implement a button that makes the recipient go through the CBUI and authorize the payment of marketplace fees for your service. The value returned by the CBUI in the tokenId parameter is used as the value for RecipientTokenId in subsequent Amazon FPS requests. The following token types can be associated with a recipient: recurring-use, multiple-use, and single-use.

Token Creation

The Co-Branded service creates a token in two cases: when a buyer successfully completes the CBUI web pages, thereby authorizing a purchase, and when a merchant authorizes the payment of marketplace fees to you. The CBUI returns to your website references to the created tokens in the tokenID parameter. This value is either used as a SenderTokenID or RecipientTokenID (depending on the implementation) in subsequent Amazon FPS requests.

The token type you create depends on the parameters included in your Co-Branded service request. This guide presents the API for each token type available in this Amazon FPS Quick Start. For more information about sending a Co-Branded service request, which can result in token creation, see "Getting Authorization."

Amazon FPS API and Co-Branded Service Requests

Amazon FPS has two production endpoints where you send requests. One is for requests involving the Amazon FPS API. These requests implement all of the financial functionality included in Amazon FPS, such as `Pay` and `Refund`. The other endpoint is for Co-Branded service requests that redirect a buyer to a series of Amazon-hosted web pages where the buyer authorizes a payment.

Amazon FPS API and Co-Branded service requests differ in the following ways:

- API requests carry out actions using the Amazon FPS web service. Co-branded service requests make the buyer interact with Amazon-hosted interface in which the buyer authorizes payments, such as when he or she authorizes the use of his or her credit card to complete a purchase.
- The response to an Amazon FPS request is an XML document. The response to a Co-Branded service request is a URI sent to a URL specified in the request.
  The requests have different endpoints, as follows.

  Amazon FPS API— https://fps.amazonaws.com

  Amazon Co-Branded service API—
  https://authorize.payments.amazon.com/cobranded-ui/actions/start

For more information about Co-Branded service requests, see "Getting Authorization."

You must make Co-Branded service requests before API requests because the Co-Branded service creates the payment token that you must use in API requests. The Co-Branded service returns pointers to those tokens in the form of token IDs.

Sandbox

Amazon FPS provides an environment called the sandbox for testing your applications. In the sandbox you can try out your requests without incurring charges or making purchases. We recommend that you test all of your requests in the sandbox before exposing them on your website.

The sandbox has separate endpoints for the Amazon FPS API Co-Branded service API.

Amazon FPS API—https://fps.sandbox.amazonaws.com

Co-Branded service—https://authorize.payments-sandbox.amazon.com/cobranded-ui/actions/start

For information about getting a sandbox account, go to Signing Up for the Amazon FPS Sandbox in the Amazon Flexible Payments Service Getting Started Guide.

Instant Payment Notification

Instant Payment Notification (IPN) is a notification mechanism that uses HTTP POST to send you immediate updates on transactions. IPN saves you the trouble of polling Amazon FPS for transaction results that complete asynchronously.

Amazon FPS sends you an IPN whenever a transaction completes, as in the following cases:

- A token is canceled successfully
- A payment or reserve succeeds
- A payment or reserve fails
- A payment or reserve goes into a pending state
- A reserved payment is settled successfully
- A reserved payment is not settled successfully
- A refund goes into a pending state
- A refund succeeds
- A refund fails
- A payment is canceled
- A reserve is canceled

> **Note**
> IPN must be configured in order to operate. If IPN is not configured, an e-mail message is the only notification made.

For information on configuring IPN, see “Setting Up Instant Payment Notification.”

Errors

Amazon FPS error results provide information about syntax errors in your requests, as well as errors that occur during the execution of your request (for example, a search that returns no results).

In the Amazon FPS API Reference, each action description contains the list of errors that can be returned.

For a list of all errors, see “Error Descriptions.”

REST Errors

If the original request to Amazon FPS used REST, in the case of an error, Amazon FPS returns an XML error response similar to the following. Errors consist of two elements: code and message.

```
Response : <?xml version="1.0" encoding="UTF-8"?>
<Response>
  <Errors>
    <Error>
      <Code>InvalidTokenId_Sender</Code>
      <Message>Sender token is not valid.</Message>
    </Error>
  </Errors>
  <RequestID>67679d8a-fd87-4e44-b063-32a69bfc3c8b</RequestID>
 </Response>
Response Code: 400>
```

The error code is a unique string that identifies the error; the error message is a human-readable description of the error. These elements are nested within an Error element. If a request generates more than one error, only the first error is reported.

Response codes are more generic errors of which the error code is a subset. For more information, see “Response Codes.”

Response Codes

Amazon FPS returns response codes in three categories so that you can easily determine how best to handle a problem:

2XX—Errors caused by mistakes in the request. For example, your request might be missing a required parameter. The error message in the response gives a clear indication of what is wrong.

4XX—Errors that are transient. These errors do not indicate a problem with Amazon FPS. So, upon receiving this error, resubmit the request.

5XX—Errors that are non-transient. These errors reflect problems with the underlying Amazon FPS web service. You will have to wait until the web service is functioning before resubmitting the request.

CE and SE Status Codes

Amazon FPS returns a status code for each of the Co-Branded service requests you make. You can receive success and failure status codes for your requests. The status codes for each of the

Co-Branded service APIs are listed in the respective topics in this guide. If you receive a caller exception (CE) or system error (SE) status code, you must handle them as described here.

CE (Caller Exception)

A caller exception (CE) error code indicates that your Co-Branded service code has an error. We assume that you will encounter any caller exceptions when you test your Co-Branded service integration (before you go live). Therefore, when a caller exception occurs, Amazon FPS displays an error message on the user interface describing the problem. If you click the provided **Continue** button, the CBUI returns you (as the test buyer) to your website (the return URL) and passes the caller exception error in the URI. You must fix the code that manages the requests to avoid receiving the error again.

Error Message	Description
CE - Caller Input Exception: The following input(s) are not well formed: [comma-separated list of input parameters]	The request parameters in the error message are not specified as mentioned in the request parameter description of the pipeline.
CE - Caller Input Exception: The following input(s) are either invalid or absent: [comma-separated list of input parameters]	The input parameters in the error messages are either incorrect or have not been specified in the request.
CE - Caller Input Exception: The following input(s) are not valid for this pipeline: [comma-separated list of input parameters]	The input parameters mentioned in the error messages should not be included for this pipeline. Please view the list of correct input parameters from the respective topic.

SE (System Error)

A system error (SE) indicates that your Co-Branded service request has temporally failed in Amazon FPS. You can retry the request again.

Business Considerations

Running a business is more than just creating a website. Creating a business involves creating policies and interacting with buyers. The business policies you make help determine the functionality you implement on your website. This section discusses such business considerations.

Amazon Payments and Your website

You can add an Amazon Payments icon to your website to let your buyers know you accept Amazon Payments. For more information, go to the Payment Marks and Graphics page on the Amazon Payments website. Also, if you have an Amazon seller account, you'll find more Amazon Payments tools, such as sample emails, payment marks, and graphics, in the Seller Central Marketing Toolkit.

Supported Payment Instruments and Currencies

Amazon FPS supports the following payment instruments:

- Amazon Payments account balance (ABT)
- Bank account debits (ACH)
- Credit cards (Visa, MasterCard, American Express, Discover, Diners Club, and JCB)

Amazon FPS allows all Amazon.com customers (U.S. and international) to use major credit cards to make payments on Amazon Payments websites. However, only US-based customers can use Amazon Payments account and bank account transfers. All transactions are conducted in U.S. dollars.

Amazon Payments Account

If buyers already have an Amazon.com account, an Amazon Payments account is automatically created, and is activated when they make their first payment on any website that accepts Amazon Payments.

If a buyer doesn't have an Amazon.com account, it's easy to create one: he or she only needs to supply an email address and a password.

Buyers can also hold a monetary balance in their Amazon Payments accounts and use this money as a payment method just like a credit card or bank account. Buyers can manage their Amazon Payments accounts through the Amazon Payments website.

Account Management

Buyers, merchants, and developers can track transactions at http://payments.amazon.com. If you prefer to programmatically track transactions, you can use the Amazon FPS Account Management Quick Start implementation to get account information, for example, for a specified period. See the Amazon FPS Account Management Quick Start Developer Guide.

> **Note**
> Buyers cannot see their account activity using their customer account on www.amazon.com.

Amazon Recipient Fees

Amazon Payments charges different fees for each of the different payment methods: credit cards, bank account debits, and Amazon Payments balance transfers. Amazon’s cost to process a payment through a bank account debit is less than the cost via credit card. Amazon's cost to process an Amazon Payments balance transfer is less still. By exposing different fees for each of these three methods, Amazon Payments can pass on savings from bank account debits and balance transfers, allowing you to save money. In each case, Amazon Payments takes on the complexity of managing security and fraud protection. Fees are assessed on a per-transaction basis and vary depending on the payment method used and the transaction. For more information, go to the FAQ on the Amazon FPS home page.

Fraud

You can feel safe and secure while your customers shop on your website. Amazon Payments is built upon Amazon's leading fraud protection technology. You could still be held liable for service chargebacks. For details, go to our User Agreement.

Disputes

We want buyers to purchase with confidence when using Amazon Payments. However, disputes between buyers and merchants do occasionally occur. When this happens, buyers should first contact the merchant directly to try to find a solution. If the parties cannot resolve their dispute, the Amazon Payments Buyer Dispute Program provides a mechanism to address the buyer's complaint using the Amazon A-Z Guarantee.

When a buyer files a dispute, Amazon will notify the seller by email. Based on the notification, the seller can choose to refund the transaction amount to the buyer or the seller can contest the dispute by providing details that prove of delivery of service or goods within 5 business days. Amazon FPS will resolve the dispute based on the information the buyer and the seller provide.

The seller should use the following tips to avoid disputes:

- Answer all buyer contacts (e.g., emails) promptly
- Be sure to deliver within the shipping estimate you provide
- Describe products accurately and provide clear images
- Keep buyers informed
- Work with buyers to resolve their negative order experiences
- Pick, pack, and ship securely. Don't skimp on packing
- Post a clear returns policy. Respond to return requests promptly with detailed instructions
- Promptly cancel any out of stock orders
- Refund as soon as possible when product defects or recalls become apparent

Amazon FPS does not provide actions to handle disputes. This section, however, addresses how to handle them.

Amazon A-z Guarantee

The Amazon A-z Guarantee applies to qualified purchases of physical goods. Therefore, the following items are not covered by the Amazon A-z Guarantee: payments for services, digital merchandise, and cash equivalent instruments (including retail gift cards). The condition of the item purchased and its timely delivery are guaranteed under the Amazon A-z Guarantee. For transactions that are not covered by Amazon A-z Guarantee, the Amazon Payments Buyer

Dispute Program still allows buyers to obtain assistance in seeking the merchant's further consideration of their complaint. Amazon Payments will attempt to resolve disputes by fostering good faith communication between buyers and merchants.

The item must be purchased from a merchant using Amazon Payments. The buyer must wait 15 days from the order date to submit a claim. From that point, the buyer has 90 days to submit a claim.

The Amazon A-z guarantee applies under the following conditions:

- If the item becomes defective more than 30 days past the shipment date and it is under warranty, the buyer must contact the manufacturer for repair or replacement. The buyer must provide all information required when submitting the claim.
- If the buyer paid by credit card, and the issuing bank has initiated a chargeback, the buyer is not eligible for coverage under the Amazon A-z Guarantee.

Buyers who pay for qualified physical goods using Amazon Payments are eligible to receive up to $2,500 of the purchase price, including shipping charges.

Amazon has built up a base of millions of satisfied customers over the years through an intense focus on being responsive to their concerns and acting quickly to resolve any outstanding problems. The vast majority of customers never need to use the Amazon A-z Guarantee reimbursement program, but for those who do, the guarantee claim gives customers a greater sense of trust and confidence in shopping from the broad range of merchants.

Amazon Buyer Dispute Program

The Amazon Buyer Dispute Program applies when the buyer has used Amazon Payments to purchase a nonphysical item or service from a merchant; and either the buyer paid the merchant for the item or service but it did not arrive; or the buyer received the item, but the item is materially different than the way the merchant described it. For more information, go to Buyer Dispute Program.

The A-z Guarantee only applies to the purchase of physical goods and does not apply to unlawful or prohibited items (including items violating the Amazon Payments Acceptable Use Policy or our User Agreement). For more information, go to the Acceptable Use Policies and Amazon Payments User Agreement.

Buyers can submit a complaint by logging into their Amazon Payments account. For disputes involving physical goods that are covered under the Amazon A-z Guarantee, we will process a submission as an A-z Guarantee claim. Buyers also can submit an A-z Guarantee claim by viewing the specific transaction details via Your Account on the Amazon Payments website. From the transaction or order details page, they can also click **"Problem with this transaction?"** or **"Problem with this order"** to file a claim.

Buyers can contact Amazon when the transaction has been resolved, but merchants are not able to withdraw claims filed by a buyer. Instead, if merchants believe that a pending claim should be revoked or canceled, they must contact buyers and encourage them to write to us. If the buyer and the seller reach a resolution after a claim check was sent, we asks buyers to contact us to make arrangements for repayment.

Chargebacks

A chargeback is a reversal of payment issued by the bank when a buyer disputes a charge. A chargeback can occur when a buyer has not received the items, has been charged multiple times for a single purchase, or is dissatisfied with the purchase and has not been able to resolve the matter with you. Chargebacks can happen only with credit card transactions.

Typically, a buyer contacts his or her bank to request a chargeback. The bank notifies the credit card association, which in turns notifies us. We work with the credit card company to resolve the chargeback. We may request information from you to dispute the chargeback with the credit card association.

Amazon FPS works with you and the buyer to resolve the chargeback. You have 5 business days to respond to the chargeback notification Amazon FPS sends you and to supply any requested information. If you do not respond within this time period, the dispute is automatically granted to the buyer.

Use the following tips to avoid chargebacks:

- Charge buyers once for a single order to avoid duplicate billing
  If you receive two or more identical orders, verify the information with the buyer
- Avoid dissatisfaction with item quality by providing a detailed description of items on your website, including specifications, measurements, and capabilities
  Other aids such as audio, video, photographs, or drawings are also helpful
- Make the shopping experience positive for your buyers:
  - Provide help when your buyers have questions or need assistance
  - Clearly explain to your buyers when their order will ship and keep them informed about the progress of their orders
  - Make sure that items are delivered promptly without damage
  - Ship items with carriers who provide online item tracking and require signatures on delivery
  - Respond promptly to email from your buyers
  - Publish your policies for cancellations and returns to avoid chargebacks
  - Refund an order when it is necessary to do so

WSDLs and Schemas

Web services involve the exchange of requests and responses between computers communicating over the Internet. To enable computers running different operating systems to communicate, the vocabulary for the communication must be established. A WSDL is a dictionary of terms that two computers can use to structure requests and responses. Schemas typically contain type definitions of the terms in the WSDL.

This section provides a brief introduction to WSDLs and schemas and also provides the location for the Amazon FPS WSDL and schema.

WSDL

A WSDL (Web Service Description Language) is an XML document that defines the operations, parameters, requests, and responses used in web service interactions. You can think of a WSDL as the contract that defines the language and grammar used by web service clients and servers. When you look at the Amazon FPS WSDL, for example, you find in it all of the Amazon FPS operation names, parameters, request and response structures.

There is not a single WSDL. Amazon FPS, for example, has many different versions of its WSDL—the latest one and all of its previous versions. Not only can one company use different versions of a WSDL, every company can use its own WSDL based on its own APIs or business metrics. For that reason, web service requests must identify the WSDL they use so the web servers know how to interpret the requests.

The latest Amazon FPS WSDL is at: https://fps.amazonaws.com/doc/2010-08-28/AmazonFPS.wsdl.

Schema

A schema is similar to a WSDL in that both are XML documents. Whereas the WSDL defines the web service language used by computers to converse, the schema defines the data types used in the WSDL.

You do not have to create schemas to use Amazon FPS. Those have already been created. It is helpful, however, to understand schemas so that you can determine the data types returned in responses.

The W3C defines the base data types, which include, for example, int, string, and float. While these data types are useful, they are not very descriptive. For example, defining every occurrence of text in an XML document as being of type string hides the differences between text that might be, for example, a paragraph versus a note. In such an application where paragraphs and notes are used, a schema would contain an extension of the string base class so that paragraph (<para>) and note (<note>) could be used as tags in XML documents.

The latest Amazon FPS schema is at: https://fps.amazonaws.com/doc/2010-08-28/AmazonFPS.xsd.

Programming Guide

The Programming Guide provides task-oriented descriptions of how to use and implement Amazon Flexible Payments Service (FPS) actions. For a complete description of Amazon FPS actions, see the Amazon FPS API Reference.

The following table describes the topics discussed in the programming guide.

> **Note**
> To perform these tasks, you must have an Amazon FPS developer account. For information about getting the account, go to Amazon Flexible Payments Service Getting Started Guide.

If you want to …	Read this section …
Enable buyers to authorize payments	Getting Authorization
Enable merchants to register on your website so they can Recipient Registration receive payment for their items sold through your website	Recipient Registration
Process the buyers' payments	Making Payments
Cancel an order or refund a payment	Order Cancellations
Test your application using the Amazon FPS sandbox	Testing Your Applications for Free
Use the Amazon FPS Sample Code	Code Samples
Create request signatures	Working with Signatures
Get notifications about transactions	Setting Up Instant Payment Notification

Amazon FPS Endpoints

Amazon FPS has four endpoints where you send requests, listed in the following table. Two are for sandbox testing of CBUI and API requests, and two are for production Co-Branded User Interface (CBUI) and API requests.

Endpoint	Purpose
https://authorize.payments-sandbox.amazon.com/ cobranded-ui/actions/start	Sandbox endpoint for Co-Branded service requests.
https://authorize.payments.amazon.com/ cobranded-ui/actions/start	Production endpoint for Co-Branded service requests.
https://fps.sandbox.amazonaws.com	Sandbox endpoint for Amazon FPS actions.
https://fps.amazonaws.com	Production endpoint for Amazon FPS actions.

Important Values to Store in Your Database

When you use Amazon FPS, there are times when you should store important information in your database. This topic describes some important values you should store.

Caller Reference

The CallerReference is a string, with a maximum length of 128 characters, which you provide that uniquely identifies the request. Any value is appropriate as long as it is unique within your system. You can also use the value to retrieve information about a transaction or to retrieve the related token (for more information, see “Co-Branded Service Requests that Don't Return”). Amazon FPS uses the caller reference value to provide request idempotency for a seven-day period (for more information, see “Resending Requests.”

> **Note**
> If you perform multiple partial refunds for a particular payment, you must provide a different caller reference value for each partial refund request.

Transaction ID

The TransactionId is a string, with a maximum length of 35 characters, which Amazon FPS creates to uniquely identify each transaction in the FPS system. The Co-Branded service doesn't return a transaction ID; only Amazon FPS does (e.g., in a Pay response).You should maintain the transaction ID in your database and associate it with your caller reference value for the order. Because of network issues, it's possible that the response to your Pay call might not reach you, so you won't have a transaction ID to store in your database. In that case you can resend the original request (within 7 days) and receive the response again (for more information, see “Resending Requests.”

Request ID

Amazon FPS returns a RequestId for each Amazon FPS API call accepted for processing. If you have a problem with a request, AWS will ask for the request ID to troubleshoot the issue. For more information about RequestId, see “Common Response Elements” in the Amazon FPS API Reference Guide.

Accepting Payments from Mobile Devices

Amazon FPS provides a seamless integration with web sites optimized for mobile devices. No special Amazon FPS coding is required. The software detects the client browser HTTP_USER_AGENT and routes the request along the appropriate CBUI pipeline. A separate pipeline is optimized for the mobile device experience.

The Amazon Payments service has been designed and developed for use within a web browser only. Our service cannot be used within a native application (including, without limitation, iOS, Android, RIM and Windows operating systems). Amazon Payments reserves the right to suspend the Payment Account of any user of our services that has implemented our Services within a native application.

> **Note**
> For optimal security, your mobile application should make use of a full browser instance (not a browser embedded within an application). It should display the browser address bar to enable customers to confirm the URL.

The user CBUI experience is managed for you. You handle all Amazon FPS requests, return URLs, and IPN notifications regardless of which client browser the customer is using.

Amazon FPS makes it easy to test your mobile client experience. For more information, see “Simulating a Mobile Client.”

Getting Authorization

Before you can issue an Amazon FPS request that charges a buyer for an item, you must get the buyer's authorization. The authorization process uses the Amazon Co-Branded service, which has a different API from the Amazon FPS web service. Some of the values returned by the Co-Branded web service, however, are required in Amazon FPS requests. When the buyer authorizes a purchase, the Amazon Co-Branded service creates a payment token, which enables the exchange of money from buyer to seller.

This section describes how to use the Co-Branded API and payment tokens. The remainder of this guide

Sending a Co-Branded Service Request

This section shows how to send a request that redirects the buyer to the CBUI. You must send a Co-Branded service request before you can use an Amazon FPS Pay or Reserve request.

These requests are typically implemented as an HTML form on your website. Your site dynamically updates the values of the Co-Branded service request parameters according to the items purchased.

To send a Co-Branded service request

1. Add up all the charges for all of the items the buyer wants to purchase, together with all taxes, shipping fees, and any additional fees (such as gift wrapping fees).
2. Use the Recipient Token API, the Recurring-Use Token API, or the Multi-Use Token API, depending on the kind of payment token you want to create (for more information about the APIs, see Co-Branded Service API Reference. For a list of the parameters common to all Co-Branded service requests, see “Common Parameters.”
   This example request is for a recurring payment for the download of a number of songs per month.

```
https://authorize.payments.amazon.com/cobranded-ui/actions/start?
```

```
callerKey=[The caller's AWS Access Key ID]
&callerReference=DigitalDownload1183401134541
&paymentReason=Monthly+download+subscription
&pipelineName=Recurring
&recurringPeriod=1+Month
&returnURL=http%3A%2F%2Fwww.digitaldownload.com%2FpaymentDetails.
jsp%3FPay
mentAmount%3
D05.00%26Download%3DMonthlySubscription%26uniqueId%3D118340113453
5
&signatureVersion=2
&signatureMethod=HmacSHA256
&signature=[URL-encoded value you generate]
&transactionAmount=5.00
```

The optional parameters vary in your request according to what the buyer purchases. For information about getting your AWS Access Key ID value, go to the Amazon Flexible Payments Service Getting Started Guide.

3. Programmatically populate this request with the parameter values based on the items the buyer is purchasing.
4. Calculate the signature and include it in the request.
   For more information about creating the correct value for signature, see “Working with Signatures.”
5. Implement the Pay Now button on your website to send this request.

Note
We recommend that you first try your Co-Branded service request in the Amazon FPS Sandbox. For more information, see “Testing Your Applications for Free.”

Recipient Registration

As a caller offering marketplace services, you must provide a way to register recipients so that they can use your services, which include uploading product information for you to display on your website and facilitating the sale of this merchandise. You must collect enough information to identify each recipient so they can get paid for products purchased by buyers. You must handle the upload of product information to your website. Amazon FPS Advanced Quick Start explains how to handle the registration of your recipients and the exchange of money from sender to recipient.

The actions Pay and Reserve, which initiate payment transactions, require parameter values called tokenIDs that identify the sender and the recipient. There is a SenderTokenId parameter, which identifies the sender (who sends the money), and a RecipientTokenId parameter, which identifies the recipient (who receives the money). Because you, the caller, send Pay requests on behalf of the others, you must obtain these tokenID identifiers.

These identifiers are generated by the Amazon Co-Branded service at the request of the Co-Branded User Interface code (CBUI), which is a series of web pages that, among other things, identifies the recipient or sender. This means that your website must have an interface that sends a Co-Branded service request for the sender and a different one for the recipient. The procedures for sending both requests are the same; the only difference is that they use different Co-Branded service APIs. The API used for recipient requests returns a value in tokenID that identifies the recipient, which is used as the value for the RecipientTokenId parameter when making a Pay or Reserve request. The API used for the sender requests returns a tokenID that identifies the sender, which is used as the value for the SenderTokenId parameter.

This section describes how to use the Co-Branded service to register the recipient.

Recipient Registration on Your Website

The following figure represents the first step in the workflow: recipient registration on your website. Merchants must register with you for the following reasons:

- The recipient must accept your business terms, in particular, the marketplace fee you will charge them.
- The recipient must be able to upload and otherwise manage the item information to your website.
- You must have a tokenID value for each recipient, so that you can pay the recipient by using the value for the RecipientTokenId parameter in a Pay or Reserve request.

Recipient Registration

Implement a recipient account system on your website.

1. For example, enable each recipient to sign in to their account with a sign-in name and password.
2. As the final stage of your registration process, implement a Register button to send a recipient Co-Branded service request. For more information, see “Recipient Token API”
3. The recipient enters the required information on the Amazon CBUI pages.
4. After confirming his or her choices, the recipient is redirected back to your website with information, such as the recipient's tokenID that you should store in your database.

Implementing the Co-Branded API to Register a Recipient

The Co-Branded API you use for registering a recipient is the Recipient Token API. For more information, see Recipient Token API.

To register a recipient

1. On your website, enable a recipient to register with you.
   In this step you collect information about the person or company using your services.
2. Direct each recipient to https://payments.amazon.com/sdui/sdui/premiumaccount to create a business payment account so they can get paid.
3. On your website, display your business policies, including your marketplace fee structure, and obtain the recipient's acknowledgment.

Your marketplace fee might include a flat fee, a percentage of the purchase price, or both. You implement your fee structure using the parameters in the Co-Branded service request.

4. Implement a button that issues a Co-Branded service request that registers the recipient. For more information, see “Sending a Co-Branded Service Request.”
5. Parse the response.

In particular, store the tokenID, which is the recipient's. You use this value for the RecipientTokenId in Pay and Reserve requests.

Making Payments

After the sender authorizes the purchase on the CBUI web pages, the URI returned contains a successful status value. Upon receiving this response, you must send a Pay request to actually initiate the transfer of money from the sender to the recipient. In the Advanced Quick Start implementation, the recipient could be you or a merchant whose e-commerce site you host.

This section describes how to make those payments.

Transacting the Payment

The sender uses the CBUI to authorize the payment. Upon the successful authorization, the CBUI redirects the sender to the URL specified by the returnURL parameter in the CBUI request. When you parse this returned URI, you check the status parameter and returnURL, among other values. If the status is one of the success values, you need to send a Pay request to start the purchase transaction. A successful Pay request immediately charges the sender's payment instrument, such as a credit card. Pay can accept all payment instrument types, including credit card, bank account debit, and Amazon Payments withdrawal.

To transact a payment using Pay

1. Obtain the list of items being purchased by the sender from your website and derive the values required for the parameters in the CBUI request.
   One of the parameters is TransactionAmount. The value for this is the total charge to the sender, including tax, shipping and any other fees. Another parameter is CallerReference.
   You generate this identifier and associate it with the payment instrument created. You can index the transactions on your database based on its value. This way, you can match the identifier on your database, CallerReference, to the identifier on the Amazon FPS database, TransactionId. For more information about the parameters, see “CBUI.”
2. Implement on your website the equivalent of a Pay Now button so that it sends the CBUI request.
   For more information about constructing the CBUI request, see “CBUI.” When the sender authorizes the payment in the CBUI, the sender is redirected to the URL specified by the returnURL parameter in the CBUI request. This URL is typically a Thank you page and one that invites the sender to keep shopping.

3. Parse the returned URI.
   In addition to the URL specified by returnURL, the URI contains parameters added by the CBUI. Those parameters include all of the parameters in the CBUI request, which are helpful for debugging purposes, a TokenId, which you must subsequently submit with the Pay request, and a status, which tells you whether or not the sender successfully authorized the payment.
4. If the status value is a success value, programmatically submit a Pay request.
   The required parameters include SenderTokenId, which maps to the TokenId you received in the CBUI response, TransactionAmount, which you entered in the CBUI request, and CallerReference, which is a value you generate that uniquely identifies the Pay transaction.

Note
The CBUI API uses the parameter, CallerReference, and the Amazon FPS API uses CallerReference. It is easier if you make both parameters equal the same value. We recommend that you make the CallerReference the same as the order ID on your website.

5. Parse the response.
   Two important values the Pay request returns are TransactionStatus, which tells whether the charge was successful against the sender's payment instrument, and TransactionId, which is the identifier Amazon Payments uses to identify this transaction. You should maintain the TransactionId in your database and associate with your CallerReference value for the transaction.
6. Upon a successful transaction, add a task to your workflow to fulfill the order.

Failed Payment Transactions

There can be times when Amazon Payments charges a sender's payment instrument and that transaction fails. There are two kinds of failure:

Failure—A transaction is canceled for non-payment reasons. For example, a transaction might be considered fraudulent and is therefore refused.

Hard decline—A financial institution refuses the transaction.

This could happen when a credit card has exceeded its maximum limit or when a credit card has expired. There is no retry after a hard decline.

You do not need to take any action in either of these cases. In the case of a failed transaction, Amazon Payments e-mails the sender and you about the transaction decline.

Repeated Pay Requests

Due to network problems, some Pay requests might not complete successfully. If this happens, it might be possible to recapture the information and resend the request. For more information, see “Resending Requests.”

Changing the Payment Instrument

It's possible that before a recurring or multi-use payment token expires, the sender must change the payment instrument associated with the token. This might happen, for example, when a credit card expires or is otherwise terminated. To handle this issue, you must enable the sender to change the payment instrument for an ongoing payment token. You do that by sending a Co-Branded service request for the Edit Token API, as described in the following process.

Process for Changing a Payment Token's Payment Instrument

1. Enable the sender to discover on your website the ongoing payment token that needs modification.
2. Enable the sender to enter the new payment instrument data, such as a credit card number.
3. Use the input to construct a Co-Branded service request for the Edit Token API. For more information, see “Sending a Co-Branded Service Request” and “Edit Token API.”
4. Implement the equivalent of a Change Payment Method button to issue the Co-Branded service request.
5. Parse the returned URI.
   In addition to the URL specified by returnURL, the URI contains parameters added by the Co-Branded service. Those parameters include all of the service response parameters, including status, which tells you whether or not the sender successfully changed the payment instrument. Part of the job of parsing the URI is evaluating the signature parameter to make sure the response has not been altered. For more information, see “Verifying the ReturnURL and IPN Notifications.”
6. Display an appropriate web page to the sender based on the value of the status parameter.

Notifications

If the sender uses an Amazon Payments account balance to make the purchase, the success or failure of the transactions occurs quickly. If, however, the sender uses a bank account withdrawal or a credit card to make the purchase, the results often take a while to complete. To be notified of the status of the transaction, you must create a web service that receives Instant Payment Notification (IPN) updates, and enable IPN updates. For more information, see “Setting Up Instant Payment Notification.”

Handling Transactions that Don't Return

When a customer buys a product on your website, their expectation is that the product will be paid for and delivered, but sometimes problems cause service requests to become lost. In those cases, it is sometimes possible to find the lost transaction.

Co-Branded Service Requests that Don't Return

After the sender finishes the CBUI pages and authorizes the payment, the service should redirect the sender to the URL you specified in the returnURL parameter in the CBUI request. There might be times, however, when the sender authorizes the payment and the redirect fails. Because you don't have the SenderTokenId, you can't charge the sender for the authorized purchase. Use the following procedure when the redirect from service fails.

After the sender finishes the CBUI pages and authorizes the payment, the service should redirect the sender to the URL you specified in the returnURL parameter in the CBUI request. There might be times, however, when the sender authorizes the payment and the redirect fails. Because you don't have the SenderTokenId, you can't charge the sender for the authorized purchase. Use the following procedure when the redirect from service fails.

To recover the SenderTokenId

1. Query your database to get the CallerReference you supplied in the Co-Branded service request for the transaction in question.
2. Send a GetTokenByCaller request with the CallerReference.
3. If there isn't an Errors element in the response, the sender authorized the payment but there was some problem with the redirect to returnURL. The response includes SenderTokenId.
   If there is an Errors element in the response, the authorization did not succeed.

Resending Requests

There are times when network problems prevent the completion of requests. This could happen both for Co-Branded service requests and for FPS API requests. Your application should periodically check for this condition, and if it occurs, retry the request.

Resending Co-Branded Service Requests

To check for Co-Branded service requests that didn't complete, whenever you send a request, put the caller reference value from the request in your database. When you get the response, store the token ID that your receive against the caller reference in your database.

About once each hour, resend any requests that don't have a sender token ID stored against the caller reference. The timing is important, because for most tokens, Amazon FPS maintains the token IDs for only three hours.

Resending Amazon FPS API Requests

Each time you send a request to the Amazon FPS API, FPS maintains the caller reference from the request for 7 days and uses it to check for duplicate requests. If you don't receive a response to an Amazon FPS API request, you can resend the exact same request within that seven-day period, and Amazon FPS will return the original response and not create a new transaction. If the first request succeeded, the second request does not charge the sender's payment instrument a second time.

In that seven-day period, if you send a request with that same caller reference value but other parameter values, Amazon FPS returns a DuplicateRequest error.

After the seven-day period, if you send the original request again (with the same caller reference and parameter values), Amazon FPS creates a new transaction.

Order Cancellations

There are times when the sender decides to cancel a recurring payment. For example, the sender might decide in the middle of a subscription to stop receiving a newspaper or magazine. This chapter describes how to cancel recurring transactions.

Canceling a Recurring Transaction

The Amazon Payments website, payments.amazon.com, enables senders to cancel recurring transactions from their user account. To provide transaction cancellation functionality on your website, use the following procedure.

To cancel a recurring transaction

1. Enable the sender to find the recurring transaction to cancel.
2. Obtain from that transaction the TokenId, which was returned in the Co-Branded service response.
3. Submit a CancelToken request with the token ID.

For more information about this action, see “CancelToken.”

The following CancelToken request cancels the recurring transaction specified by the TokenId.

```
https://fps.sandbox.amazonaws.com?
Action=CancelToken
&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
&ReasonText=MyWish
&Signature=IZD9O%2FWGqhkzO%2FdLTQ7Tn8KUAmtZXqIEg6gypwkGeWQ%3D
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2009-10-07T08%3A46%3A37.156Z
```

```
&TokenId=D739IT9TMC4FK9KB56PDKJWAQGXDZ3B8X3SJNGVH3UEF5GQ7XAQZMEIL
4OGEZKGX
&Version=2008-09-17
```

Refunding a Recurring Transaction

Amazon FPS gives you the option of refunding an unused portion of a recurring payment that the sender paid for but will not use. This might happen if, for example, the sender pays membership fees twice annually but must cancel only three months into the membership. The Refund action has an optional parameter, RefundAmount, which enables you to refund all or only a portion of the transaction amount a sender paid.

The Amazon Payments website, payments.amazon.com, enables senders to refund transactions from their user account. To provide transaction refunding functionality on your website, use the following procedure.

To refund a transaction

1. Enable the sender to find the transaction to refund.
2. Obtain from that transaction either the TransactionId or the CallerReference.
3. Submit a Refund request using one of those values.

For more information about this action, see “Refund.”

The following Refund request refunds the transaction specified by the CallerReference and TransactionId.

Other Reversals and Issues

In the e-commerce world, there are other types of transaction reversals, including chargebacks and claims. Amazon Payments does not provide actions to handle those specific activities. So, those implementations are left to you on your website. For more information, see “Business Considerations.”

Testing Your Applications for Free

Amazon FPS provides a sandbox environment that you use to test your applications. In the sandbox you can try out your applications without incurring charges or making purchases. We recommend that you test all of your requests in the sandbox before exposing them on your web site.

The Amazon FPS Sandbox enables you to:

- Make Amazon FPS web service and Co-Branded service requests
- Make Pay requests to transfer money

- Use credit cards and bank accounts in your test transactions without any prior verification and without incurring charges
- Simulate errors

You can simulate certain errors that could appear in a real transaction. This simulation can help you test the error handling capabilities in your application.

For information about signing up for an Amazon FPS Sandbox account, go to the Amazon Flexible Payments Service Getting Started Guide. For more information about the Amazon FPS Sandbox, go to https://payments-sandbox.amazon.com.You must be logged in to view this page.

Sandbox Endpoints

Sandbox endpoints are different from Amazon FPS production endpoints. The Amazon FPS Sandbox endpoints are as follows:

- **Amazon FPS API—** https://fps.sandbox.amazonaws.com
- **Amazon Co-Branded service—**https://authorize.payments-sandbox.amazon.com/cobranded-ui/actions/start
- **Central FPS Sandbox Resource page—** http://docs.amazonwebservices.com/AmazonFPS/latest/SandboxLanding/index.html

Sandbox Use

You can test the following user experiences in the sandbox:

- Registering for a business or personal account via a Co-Branded service request
- Depositing funds into a test account's Amazon Payments account using a `Pay` request
- Checking the account balance for a test account
- Checking the activity for a test account
- Tracking the cumulative effect of a series of Pay calls. While you can't adjust the time/date of the call, you can check that the values change as expected in your test account(s) with each transaction.

Simulating a Mobile Client

You can easily test the CBUI pipeline that your customers experience when they use their mobile devices. Amazon FPS uses the value of the client browser's HTTP_USER_AGENT to route the request along the appropriate pipeline. If you set your development environment to report a value for HTTP_USER_AGENT reported by a mobile device, Amazon FPS will invoke the mobile pipeline.

For example, the following value simulates an Apple iPhone 3G version 2.1, with Safari 3.1.1:

```
Mozilla/5.0 (iPhone; U; CPU iPhone OS 2_1 like Mac OS X; en-us)
AppleWebKit/525.18.1 (KHTML, like Gecko)
Version/3.1.1.1 Mobile/5F136 Safari/525.20
```

Error Simulation

The sandbox accepts any random number as a credit card and token ID in Pay and Reserve requests. However, you can simulate a variety of declines that occur by using specific token IDs and amounts in the Amazon FPS Sandbox, as shown in the following tables.

The following table shows the errors you can simulate by entering specific SenderTokenId values.

Error	SenderTokenId Value

Closed account	Z1LGRXR4HMDZBSFKXELA32KZASGWD8IHMHZCK4DETR784LDLD1GMFW4P3WT8VTGX
Email address not verified	E3FR7BARJV3PB631PMKV74PGKCJLBHI1Q1KMQN7BJ2JJICPDKN3N1CJIKFZ8D7NN
Suspended account	H216UECZ8ZM1G8G4QA3V7RKF8JDFZ9SI3SJAFSGUKBBNDHX1NVM8GUQRZNRNAHER

The following table shows the errors you can simulate by entering specific RecipientTokenId values. These token IDs are relevant only in marketplace environments (where the caller is not the recipient).

Error	RecipientTokenId Value
Closed account	P1LL7A1LHK935DBGI5NAYCXOCLVEBHBNIU7PBXBAMRKKNLDEPI8M3MUSLZT2VANZ
Email address not verified	C4LGSEMXN11FTUXZ2X2C7QVFHN5DVBGQJNF17AIQXXXQSX4DRG4KJFCN2KRFUUZI
Suspended account	R3VK49XVGCAZTJSXKN7ZSBHPMFGKM5VEEQTXGMVE8CFUZ2G5RLLMAB4J6TQRL6BU

With the Amazon Payments developer sandbox, you can force an error by placing certain decimal values in the amount. The following table details the values.

Error	Error Forced	Simulation
The amount includes a decimal value between .60 and .69	Temporary Decline	Occurs when a downstream process is not available.
The amount includes a decimal value between .70 and .89.	Payment Error	Insufficient funds

> **Note**
> If you want your test transaction to be a success, avoid using amount values which contain decimal values between .60 and .89. For example, the following amounts all force errors: 0.61, 123.6522, 1.79. The following amounts do not force an error: 0.16, 123.56, 8.97.

Testing Signatures

You can easily test your signature creation code using any of the examples in Amazon FPS API Reference. Each example contains a signature calculated from the values in the rest of the example.

1. Copy any one of the sample query request examples from among the Actions in Amazon FPS API Reference.
2. Remove the HTTP verb (GET or POST) and the URI from your copy. Also remove the explicit '\n' characters.

3. Remove the line with the Signature parameter from your copy.
4. Create a signature using the instructions in Generating a Signature.
5. Compare the output from your signature creation code with the value you removed from the HTML example. They should be identical.

Migrating your Application from the Sandbox to Production

When your application is running correctly in the sandbox, you need to do the following to switch it to the production environment:

Launch Process

1. Change the Amazon FPS sandbox endpoint to the Amazon FPS live endpoints as listed in the following table:

Application	Endpoint
Co-Branded UI Pipeline	https://payments.amazon.com/sdui/sdui/managecobranding
Amazon FPS web service requests	https://fps.amazonaws.com/
Amazon Payments Account Management UI	https://payments.amazon.com/

2. If your application is set up to receive IPN notifications, set its IPN URL at https://payments.amazon.com/sdui/sdui/managecobranding.
3. Please ensure you have specified your co-branding URL on production using the form at https://payments.amazon.com/sdui/sdui/managecobranding.
4. Please ensure that the rest of your account settings are current at https://payments.amazon.com/sdui/sdui/accountsettings.
5. For a marketplace application, make sure you register for that option when you register the application in the production environment. If you did not select the option when you registered the application, you can file a contact-us request at https://payments.amazon.com/sdui/contactus.

You can use the same credentials to sign your requests as long as your Amazon Payments Developer account on both Sandbox and Production are linked to the same email address and password.

Working with Signatures

This section provides detailed explanations for some of the tasks required to generate a signature. A signature is required for every request. For sample code for generating signatures, see “Code Samples.”

Generating a Signature

Web service requests are sent using SSL (HTTPS) across the Internet and are subject to tampering. Amazon FPS uses the signature to determine if any of the parameters or parameter values were changed in a web service request. Amazon FPS requires a signature to be part of every request.

To create the signature

1. Create the canonicalized query string that you need later in this procedure:
    a. Sort the UTF-8 query string components by parameter name with natural byte ordering. The parameters can come from the GET URI or from the POST body (when Content-Type is application/x-www-form-urlencoded).
    b. URL encode the parameter name and values according to the following rules:
        - Do not URL encode any of the unreserved characters that RFC 3986 defines. These unreserved characters are A-Z, a-z, 0-9, hyphen (-), underscore (_), period (.), and tilde (~).
        - Percent encode all other characters with %XY, where X and Y are hex characters 0-9 and uppercase A-F.
        - Percent encode extended UTF-8 characters in the form %XY%ZA....
        - Percent encode the space character as %20 (and not +, as common encoding schemes do).

        > **Note**
        > Currently all AWS service parameter names use unreserved characters, so you don't need to encode them. However, you might want to include code to handle parameter names that use reserved characters, for possible future use.

    c. Separate the encoded parameter names from their encoded values with the equals sign (=) (ASCII character 61), even if the parameter value is empty.
    d. Separate the name-value pairs with an ampersand (&) (ASCII code 38).
2. Create the string to sign according to the following pseudo-grammar (the `"\n"` represents an ASCII newline).

```
StringToSign = HTTPVerb + "\n" +
ValueOfHostHeaderInLowercase + "\n" +
HTTPRequestURI + "\n" +
CanonicalizedQueryString <from the preceding step>
```

3. Calculate an RFC 2104-compliant HMAC with the string you just created, your Secret Access Key as the key, and SHA256 or SHA1 as the hash algorithm. For more information, go to http://www.ietf.org/rfc/rfc2104.txt.
4. Convert the resulting value to base64.
5. Use the resulting value as the value of the Signature request parameter.

> **Important**
> The final signature you send in the request must be URL encoded as specified in RFC 3986 (for more information, go to http://www.ietf.org/rfc/rfc3986.txt). If your toolkit URL encodes your final request, then it handles the required URL encoding of the signature. If your toolkit doesn't URL encode the final request, then make sure to URL encode the signature before you include it in the request. Most importantly, make sure the signature is URL encoded only once. A common mistake is to URL encode it manually during signature formation, and then again when the toolkit URL encodes the entire request.

About Signature Version 2

For inbound requests, signature version 2 signing uses the entire request uri as the basis for the signature, and encryption is based on the unique security credentials for your account.

For outbound notifications, signature version 2 provides the Amazon FPS action, VerifySignature, which enables you to securely check a response using a server-side call.

> **Important**
> The original implementation of signature version 2 supported client-side signature validation using PKI. Client-side signature validation was deprecated on November 3rd, 2009, and as of 10 February, 2011 it is no longer supported. If you have been using client-side signature validation, you must switch to server-side validation using the FPS action VerifySignature.

Signature version 2 supports AWS access key rotation, further enhancing the security of your button content. For more information, see “Access Key Rotation.”

> **Important**
> The previous method for signing (signature version 1) was deprecated on November 3rd, 2009, and as of 10 February, 2011 it is no longer supported. Whenever you sign a request with your access keys, you must now use signature version 2.

Verifying the ReturnURL and IPN Notifications

Amazon Simple Pay sends you outbound notifications for both the ReturnURL and IPN notification. For the ReturnURL, it is in the form of GET data, and for IPN notification, it is POST data. When you handle these notifications, we recommend you validate the signature to ensure the notification originated from Amazon Payments.

Signature version 2 security enables you to verify the signature of the response using a server-side call to the VerifySignature FPS Action.To use it, modify your returnUrl and ipnUrl pages to parse the notification. From those components, you assemble the relevant parameters for VerifySignature and sign it like any other request. The result of the call is either Success, meaning the response is valid, or Failure, indicating the response is suspect.

For more information on VerifySignature, see “VerifySignature.” In addition, you can use the validation samples to assist creating your own validation pages. For more information, see “Understanding the IPNAndReturnURLValidation Sample.”

> **Important**
> The original implementation of signature version 2 supported client-side signature validation using PKI. Client-side signature validation was deprecated on November 3rd, 2009, and as of 10 February, 2011 it is no longer supported. If you have been using client-side signature validation, you must switch to server-side validation using the FPS action VerifySignature.

Access Key Rotation

If you decide that it is necessary to change your access keys, the security credentials page (available from your account page at the Amazon Web Services website at http://aws.amazon.com) enables you to create a second set, and allows you to activate and deactivate the sets independently.

With both sets active, you can propagate the new set to your applications over time, maintaining the high security that signing provides. Since both sets are valid, you don't have to take your entire application down to incorporate the new keys. When the distribution is complete you can deactivate the old set.

> **Note**
> You can have two sets of keys only. Both, one, or neither of them can be active.

Soft Descriptor Customization

Credit card companies allow a descriptive string on credit card statements that identify a purchase. For example, AMZN PMTS appears on credit card statements to identify purchases made using Amazon FPS.

Typically, most banks support a 19 character string. To give you more flexibility to identify yourself on credit card statements, Soft Descriptor Customization lets you modify information sent to the payment processor. You can use Soft Descriptor Customization in the following ways.

- Create a static string in your account settings

> **Note**
> In marketplace applications the soft descriptor of the **recipient's** account settings are used when the soft descriptor type is Static.

- Create a dynamic string when you process the payment
- Specify whether the recipient or caller customer service number is sent to the payment processor

How the Soft Descriptor Works

Option 1 You specify the soft descriptor string in your account level settings. Amazon FPS passes ASI* plus the soft descriptor (in upper case) to the payment processor. For example, ASI*DIGITALDOWNLOAD appears on the statement.

Option 2 You supply a sender description when processing the payment and specify Dynamic as the soft descriptor type. Amazon FPS passes a 19 character string, which consists of ASI* plus the first 15 characters of the sender description (in upper case) to the payment processor. For example, if AMAZON FPS GIFT is the sender description then ASI*AMAZON FPS GIFT appears on the statement.

Option 3 The default Amazon FPS descriptor Amazon Payments appears on the statement if you do not specify a soft descriptor string in your account level settings or a sender description in your call to Amazon FPS.

> **Note**
> Prior to June of 2010, the values were "AMZ*" instead of "ASI*" and "AMZN PMNTS" instead of "Amazon Payments."

SoftDescriptorType

Use the SoftDescriptorType in the DescriptorPolicy to specify static or dynamic soft descriptors. When you make a call to Pay, FPS checks the SoftDescriptorType parameter in the DescriptorPolicy. If you specify the parameter as Static, or do not specify a type, the soft descriptor in your account level setting is sent to the payment processor.

> **Note**
> In marketplace applications, the soft descriptor of the recipient's account settings are used when the soft descriptor type is Static.

If you need a dynamic soft descriptor string, you must specify a sender description in the Pay action. You must also specify Dynamic as the soft descriptor type. Following the soft descriptor standard, the FPS soft descriptor consists of 19 characters beginning with the string AMZ* , followed by the first 15 characters of the sender description. You can use numbers, letters, or spaces in your soft descriptor as long as the descriptor doesn't begin or end with a space.

Special characters are not allowed in the soft descriptor string. Amazon FPS returns an error if you don't include a sender description for the dynamic string.

To create a static soft descriptor

1. Log in to your Amazon Payments account at http://payments.amazon.com.
2. Point to Edit My Account Settings.
3. Click **Change My Business Settings**.
4. Enter the soft descriptor in the text box.

You can use numbers, letters, or spaces in your soft descriptor as long as the descriptor doesn't begin or end with a space. Special characters are not allowed in the soft descriptor string.

CSOwner

In scenarios like marketplace applications, the caller and recipient are different parties. You can specify the customer service number that a customer sees on his credit card statement with the CSOwner parameter. When you make a call to the Pay action. FPS checks the CSOwner parameter. If you specify the value of the parameter as Recipient, or do not specify any value, the recipient's customer service number is determined from account information and sent to the payment processor. If you specify Caller as the value of the CSOwner parameter, the caller's customer service number is determined from account information and sent to the payment processor.

> **Note**
> The soft descriptor and owner are passed to a Reserve operation are passed to the corresponding Settle operation. The original soft descriptor and owner passed to the Pay or Reserve operations are passed to a corresponding Refund operation.

Setting Up Instant Payment Notification

When the sender uses ABT (Amazon Payments Balance Transfer) to pay for a purchase, the purchase is approved or denied synchronously, which means that processing stops until the Pay call returns, and this happens relatively quickly. When the sender uses ACH (bank account withdrawal) or a credit card, the purchase is asynchronous, which means that it can take much longer to succeed or fail.

Since you cannot easily determine when asynchronous transactions complete, Amazon FPS has created a notification service called Instant Payment Notification (IPN) that uses HTTP POST to notify you when the following asynchronous transactions occur:

- A payment or reserve succeeds
- A payment or reserve fails
- A payment or reserve goes into a pending state
- A reserved payment is settled successfully
- A reserved payment is not settled successfully
- A refund succeeds
- A refund fails
- A refund goes into a pending state
- A payment is canceled

- A reserve is canceled
- A token is canceled successfully

> **Note**
> IPN must be configured in order to operate. If you do not configure IPN, only email notifications will be sent.

IPN is a simple way to process updates from Amazon FPS and has the following benefits compared to other notification mechanisms:

- Easy implementation (compared to polling for updates)
- Robust delivery mechanism
- Robust to changes in message parameters
- Simple message structure

> **Tip**
> If you have signed up for IPN and do not receive notifications, verify the URL you provided in your account settings. IPN will try for a day to deliver a notification before it gives up.

Setting Up IPN Preferences

To receive IPN notifications, you need to set up a web service that receives IPN notifications from Amazon FPS and register the URL of that web service in your Amazon FPS developer account on http://payments.amazon.com.

If you decide to use IPN, you must sign in to your Amazon Payments account, and use the following procedure to enter the URL for your web server. Once you sign up for IPN, notifications are sent to your server.

To configure your developer account so that you receive IPN messages

1. Log in to the Amazon Payments website at http://payments.amazon.com.
2. Click **Edit My Account Settings**.
3. The Edit My Account Settings page displays.
4. Click **Manage Developer and Seller Preferences**.
5. The Manage Developer and Seller Preferences page appears.
6. Enter the URL for your IPN server in the URL for Instant Payment Notification text box.

Receiving IPN Notifications

Amazon FPS uses HTTP POST to send IPN notifications to the URL registered in your Amazon Payments developer account. Use the following process to create a script that handles IPN notifications.

Tip
If your IPN receiving service is down for some time, it is possible that our retry mechanism will deliver the IPNs out of order. If you receive an IPN for TransactionStatus (IPN), as SUCCESS or FAILURE or RESERVED, then after that time ignore any IPN that gives the PENDING status for the transaction.

Setup Process for a Script to Receive IPN

1. Set up your web server to receive the HTTP POST IPN notifications on one of the following ports: 8080, 80 [http], 8443, or 443 [https].
2. Write a program that parses the IPN elements (for a list of the elements, see “Common IPN Response Elements”).
3. Write your program so that it verifies the signature value sent in the IPN to make sure Amazon FPS sent the IPN. For more information, see Verifying the ReturnURL and IPN Notifications, below.
4. Write your program to use the returned elements to notify you of the IPN-related transactions.

How To Verify the IPN Signature

You must ensure that the IPN indeed came from Amazon Payments. You can do this by verifying the value of the signature parameter contained in the response. IPN responses contain the components you need to validate with server-side signature verification. For more information, see “Verifying the ReturnURL and IPN Notifications.”

You can use the IPNAndRuturnURLValidation sample to assist creating your own IPN validation page. For more information, see “Understanding the IPNAndReturnURLValidation Sample.”

Common IPN Response Elements

These IPN response elements are common to most types of transactions. For a list of IPN response elements for marketplace transactions, see “IPN Responses for Marketplace Transactions.”

Name	Description
addressFullName	Full name of the buyer/sender. Type: String
addressLine1	Sender's address (first line). For IPN, this element is returned only if the value has been updated with Amazon. Type: String
addressLine2	Sender's address (second line). For IPN, this element is returned only if the value has been updated with Amazon. Type: String

Name	Description
addressState	Sender's state. For IPN, this element is returned only if the value has been updated with Amazon. Type: String
addressZip	Sender's post code. For IPN, this element is returned only if the value has been updated with Amazon. Type: String
addressCountry	Sender's country. For IPN, this element is returned only if the value has been updated with Amazon. Type: String
addressPhone	Sender's phone number. For IPN, this element is returned only if the value has been updated with Amazon. Type: String
buyerEmail	Sender's email address. **Note** The buyerEmail element is not returned when the recipient is not the caller (i.e., marketplace transactions). Type: String Size: 65 bytes
buyerName	Sender's name. Type: String Size: 128 bytes
certificateUrl	A url specifying the location of the certificate used for signing the response. Type: String Max Size: 1024 bytes
customerEmail	Customer's email address. Type: String Size: 65 bytes
customerName	Buyer/Sender Full Name. Type: String Size: 128 bytes
dateInstalled	If the notificationType element (below) is TokenCancellation, this element contains the date the token was installed. Type: String Size: 30 bytes
isShippingAddressProvided	If the IPN results include address updates, this element contains TRUE. Otherwise this element is not present in the response. Type: String
Operation	The name of the payment action, also called an operation, used for this transaction. For example, PAY for the action Pay. Type: String Max Size: 20 bytes
notificationType	Notification type may be either TokenCancellation or

Name	Description
	TransactionStatus Type: String Size: 20 bytes
paymentMethod	The payment method used by the sender. For more information, see the “IPN values in PaymentMethod.” Type: String Size: 20 bytes
paymentReason	Reason for payment. Type: String
recipientEmail	Recipient's email address. **Note** As a security precaution, you should always check that the recipient email is the same as the one in your original request. Type: String Size: 65 bytes
recipientName	Recipient's name. Type: String Size: 128 bytes
Signature	The encoded string the caller uses to verify the IPN. Amazon Payments calculates the signature using the elements in the returnURL. The merchant must have manually signed the request. For more information, see “Handling the Receipt of IPN Notifications.” We recommend that you always verify the signature using the method in How to Verify the IPN Signature. Type: String Size: 512 bytes
signatureVersion	A value that specifies the Signature format. Type: Integer Valid Values: 2
signatureMethod	A value that specifies the signing method. Type: String Valid Values: HmacSHA256 (preferred) and HmacSHA1.
tokenId	If notificationType is TokenCancellation, this element contains the ID of the cancelled token. Type: String Size: 65 bytes
tokenType	If notificationType is TokenCancellation, this element contains the type of the canceled token. Type: String Size: 20 bytes
transactionAmount	Specifies the amount payable in this transaction; for example, USD 10.00. Type: String Size: 30 bytes
transactionDate	The date when this transaction occurred, specified in

Name	Description
	seconds since the start of the epoch. Type: Long Size: 40 bytes
transactionId	Unique ID generated by Amazon FPS for this transaction. This element is returned if the transaction was accepted by Amazon FPS. Type: String Size: 35 byte
transactionStatus	Specifies the status of the transaction. For more information, see “TransactionStatus (IPN).” Type: String

IPN Responses for Marketplace Transactions

The following IPN response elements are returned only for marketplace transactions.

IPN Marketplace Transaction Elements

Name	Description
buyerName	Sender's name. Type: String
Operation	The name of the payment action, also called an operation, used for this transaction. For example, PAY for the action Pay. Type: String Max Size: 20 bytes
paymentMethod	The payment method used by the sender. For more information, see the “IPN values in PaymentMethod.” Type: String
paymentReason	Reason for payment. Type: String
recipientEmail	Recipient's email address. Type: String
recipientName	Recipient's name. Type: String
referenceId	If you specified a referenceId in the button creation form, Amazon Payments returns the referenceId to you. Type: String
Signature	The encoded string the caller uses to verify the IPN. Amazon Payments calculates the signature using the elements in the returnURL. The merchant must have manually signed the request. For more information, see “Handling the Receipt of IPN Notifications.”We recommend that you always verify the signature using the method in How to Verify the IPN Signature. Type: String
Status	Specifies the status of the transaction. For more information, see “TransactionStatus (IPN).” Type: String
transactionAmount	Specifies the amount payable in this transaction; for example, USD 10.00. This element is not being returned in the current version. Type: Double
transactionDate	The date when this transaction occurred, specified in seconds since the beginning of the epoch. Type: Long
transactionId	Unique ID generated by Amazon FPS for this transaction. This element is returned if the transaction was accepted by Amazon FPS. Type: String

Email Notification Templates

Many transactions generate email messages from Amazon Payments, sent to either the buyer, seller, or web site owner in the case of a marketplace transaction. Transaction details are listed in the body of the email message. The content of the email message sent out depends on the transaction and its status. The table here defines the templates that are used, and provides a link to an example message for each.

Email Template Name	Description
MultiPaymentFailure	Payment failure from a multi use token (payment authorizations)
MultiPaymentSuccess	Payment success from a multi use token (payment authorizations)
MultiuseTokenCancel	Multi-use token has been canceled
MultiuseTokenInstall	Mulit-use token has been installed
RecurringPaymentFailure	Recurring payment has failed
RecurringPaymentSuccess	Recurring payment has succeeded
RecurringPaymentSuccessACH	Recurring ACH payment has succeeded
RecurringTokenCancel	Recurring token is canceled
RecurringTokenInstall	Recurring token is installed

Amazon FPS API Reference

This section provides reference material for the Amazon FPS API.

The current version of the Amazon FPS API is 2010-08-28.

The WSDL is located at https://fps.amazonaws.com/doc/2010-08-28/AmazonFPS.wsdl.

The schema is located at https://fps.amazonaws.com/doc/2010-08-28/AmazonFPS.xsd.

> **Note**
> To use the Amazon FPS API, you must have an Amazon FPS developer account. For information about getting the account, go to Amazon Flexible Payments Service Getting Started Guide.

Actions

This section describes the actions available with Amazon FPS Advanced Quick Start.

Cancel

Description

The Cancel action cancels a reserved or pending transaction. Once the transaction is canceled, you can't then settle it. You also can't use Cancel on a completed transaction. After a transaction is completed, you can do a refund if you want to reverse the order.

If the sender's credit card was in a reserved state, it is not part of this action to make sure the reserved status is removed. There is no action required to remove the reserved status, it will automatically expire after 7 days.

Request Parameters

Parameter	Description	Required
Description	Describes the reason for cancellation. Type: String Default: None	No
OverrideIPNURL	Specifies the URL to receive the Instant Payment Notification (IPN) for the request. The IPN URL set with this parameter takes precedence over the IPN URL set in your account. Type: String Default: None	No
TransactionId	Specifies the transaction that needs to be canceled. This ID should have been returned by	Yes

Parameter	Description	Required
	Amazon in a prior Pay or Reserve call. Type: String Default: None Constraint: Max size = 35 characters	

For in your request you must include parameters that are common to all requests. For more information, see “Common Request Parameters.”

Response Elements

Element	Description
TransactionId	The ID of the completed transaction. It is the same as the TransactionID provided in the request. Type: String Size: 35 Bytes
TransactionStatus	The status of the cancellation request. Type: TransactionStatus Size: 20 Bytes

Responses also include elements common to all responses. For more information, see “Common Response Elements.”

Errors

This action can return the following errors:

- AccessFailure
- AccountClosed
- AuthFailure
- ConcurrentModification
- InternalError
- InvalidClientTokenId
- InvalidParams
- InvalidTransactionState
- SignatureDoesNotMatch

Sample REST Request

```
https://fps.sandbox.amazonaws.com?
Action=Cancel
&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
&Description=MyWish
&Signature=yOedrTuiMoMrKt8SwugDDnfd0nydyoX9uPq1H1SUCl4%3D
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2009-10-06T09%3A14%3A58.796Z
&TransactionId=14GKI1SKSR1V6DO1RCCB32RBR6KLODMGQUD
&Version=2008-09-17
```

Sample Response to REST Request

```
<CancelResponse xmlns="http://fps.amazonaws.com/doc/2008-09-17/">
  <CancelResult>
    <TransactionId>14GKI1SKSR1V6DO1RCCB32RBR6KLODMGQUD</
TransactionId>
    <TransactionStatus>Cancelled</TransactionStatus>
  </CancelResult>
  <ResponseMetadata>
    <RequestId>6fe4b755-a328-419d-8967-e1d3b43779fc:0</RequestId>
  </ResponseMetadata>
</CancelResponse>
```

Sample IPN Success Notification to Rest Request

```
--------------------------
transactionId: 14GKI1SKSR1V6DO1RCCB32RBR6KLODMGQUD
statusMessage: The transaction was explicitly cancelled by the
caller.
transactionDate: 1254820475
signatureVersion: 2
signatureMethod: RSA-SHA1
buyerEmail: new_premium@amazon.com
notificationType: TransactionStatus
callerReference: CallerReference08
transactionAmount: USD 1.00
transactionStatus: CANCELLED
operation: RESERVE
recipientEmail: test-caller@amazon.com
buyerName: Test Business
signature:
jWDbBxtEhw2rQEyMeEXcpWCgoZvm8rjLEnmg38oYoPPR7NbMGgmMA9/5CDjt9Q/FM
ktKM
bARXnZF
```

```
YTzHj3YOKiAM3vxI0zT1oTiSdBx1KBRFzK7mauxxlQv5BYxjFX+R5cl+keCaT2nQy
rp3agdrIIp5
MZ5Oy9dBuYMwMFWXoZZor90EidD23hBdZSOOzQRUdzKaKJsF14RQVrKcf5pDCs1Ha
B6LBKbATaNT
RSxxrviIXy9JcWRQhJwzcc1H6cFOJDpNFSJ03b0Z94eL/XNu9BU7bT4KRWb+OHF0P
n53yf4zyBT9
jTD+94WeujCxwE2rF0j5+brmXp/+Sn/RccDG7w==
recipientName: Test Business
paymentMethod: CC
certificateUrl:
https://fps.sandbox.amazonaws.com/certs/090909/PKICert.pem
paymentReason: Reserve
statusCode: Cancelled
--------------------------
```

CancelToken

Description

The CancelToken action cancels a multi-use or recurring payment token. You can use this action at any time during the life of the token. After this request completes successfully, Amazon FPS stops all further payments that use the specified token. There is no way to reactivate a canceled token.

You can only cancel tokens that you have created.

Request Parameters

Parameter	Description	Required
OverrideIPNURL	Specifies the URL to receive the Instant Payment Notification (IPN) for the request. The IPN URL set with this parameter takes precedence over the IPN URL set in your account. Type: String Default: None	No
ReasonText	Reason for canceling the payment token. Type: String Default: None	No
TokenId	Specifies the token to cancel. You should have stored this value when it was returned as part of the response to the Co-Branded service request. Type: String Default: None	Yes

In your request you must include parameters that are common to all requests. For more information, see “Common Request Parameters.”

Response Elements

The response for this API includes only elements common to all responses. For more information, see “Common Response Elements.”

Errors

This action can return the following errors:

- AccessFailure
- AccountClosed
- AuthFailure
- ConcurrentModification
- DuplicateRequest
- InternalError
- InvalidClientTokenId
- InvalidParams
- InvalidTokenId
- SignatureDoesNotMatch
- TokenAccessDenied

Sample REST Request

```
https://fps.sandbox.amazonaws.com?
Action=CancelToken
&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
&ReasonText=MyWish
&Signature=IZD9O%2FWGqhkzO%2FdLTQ7Tn8KUAmtZXqIEg6gypwkGeWQ%3D
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2009-10-07T08%3A46%3A37.156Z
&TokenId=D739IT9TMC4FK9KB56PDKJWAQGXDZ3B8X3SJNGVH3UEF5GQ7XAQZMEIL
4OGEZKGX
&Version=2008-09-17
```

Sample Response to REST Request

```
<CancelTokenResponse xmlns="http://fps.amazonaws.com/doc/2008-09-
17/">
  <ResponseMetadata>
    <RequestId>a10e0ad6-148f-4afe-8bcd-e80a2680793d:0</RequestId>
  </ResponseMetadata>
</CancelTokenResponse>
```

Sample IPN Notification to REST Request

```
--------------------------
signatureVersion: 2
signatureMethod: RSA-SHA1
customerEmail: test-caller@amazon.com
tokenId:
D739ATGTM94QK9NBU6P4KDWACGXDZ8BVX3TJHGVP3XEFMGE7XVQTMEIL4OGFZMGP
callerReference: CallerReference19
notificationType: TokenCancellation
signature:
flxZtuxk3jb0Ww4g4duMjx1s8EQnIC7kPHqKKu0t4trp1/8ZU6ohtm9V1xB1mdxDn
J37lpy
fL7rp
wE5tiKjJ8agm1OzPjp9rwEVOEMcdscopTVhh9AG2HTNGyWyyaRlIPlXiV3mpPyMrt
tLiOkrYB8ak
YZ9fMbXUB9gKzMVzNhh58auyD/weMV/WIX3DDSJslsp0kg6frHv5F5CYrprwv4S+c
XQxXdgJlRC3
UJO8bH68bwlFnyyzPz4+TnbB5xMDatpwkBOFCWO5+tmwlwJHyAUa7z6XJgwj27YII
jFSJolWLKwK
iZHqPNYNjKHE190sQMQBLHcnkZeexig6wYHK5w==
tokenType: SingleUse
dateInstalled: Oct 8, 2009
certificateUrl:
https://fps.sandbox.amazonaws.com/certs/090909/PKICert.pem
customerName: Test Business
--------------------------
```

GetRecipientVerificationStatus

Description

GetRecipientVerificationStatus enables you to test that the intended recipient has a verified Amazon Payments account before you present the payment option for that seller or recipient on your website. The RecipientVerificationStatus return parameter enables you to determine whether the account is unlimited in the amount of money it can receive.

Request Parameters

Parameter	Description	Required
RecipientTokenID	The recipient token returned by the Co-branded user interface. Type: String	Yes

In your request you must include parameters that are common to all requests. For more information, see “Common Request Parameters.”

Response Elements

Element	Description
RecipientVerificationStatus	Status of the verification. Type: RecipientVerificationStatus

Responses also include elements common to all responses. For more information, see “Common Response Elements.”

If a response status is failure, the Errors element includes an error code that identifies the source of the failure. If the response status is success, the elements listed in the preceding table are returned.

Errors

This action can return the following errors:

- InternalError
- InvalidAccountState
- InvalidParams
- InvalidTokenId
- TokenNotActive

Sample REST Request

```
https://fps.sandbox.amazonaws.com?
Action=GetRecipientVerificationStatus
&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
&RecipientTokenId=09DG234OGD
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2008-08-06T13%3A00%3A01Z
&TokenId=254656Example83987
&Version=2008-09-17
&Signature=[URL-encoded signature value]
```

Sample Response to REST Request

```
<GetRecipientVerificationResponse
xmlns="http://fps.amazonaws.com/doc/2008-09-17/">
  <GetRecipientVerificationResult>
    <RecipientVerificationStatus>
VerificationComplete
    </RecipientVerificationStatus>
  </GetRecipientVerificationResult>
  <ResponseMetadata>
```

```
    <RequestId>197e2085-1ed7-47a2-93d8-d76b452acc74:0</RequestId>
  </ResponseMetadata>
</GetRecipientVerificationResponse>
```

GetTokenByCaller

Description

The GetTokenByCaller action returns the details about the token specified by a tokenId or CallerReference.The CallerReference is the value you passed in the Co-Branded service request, whereas the tokenId is the value you received in the Co-Branded service response.

Request Parameters

Parameter	Description	Required
CallerReference	A value you provide that uniquely identifies the request. For more information, see “Important Values to Store in Your Database.” Type: String Default: None Constraint: Max size = 128 characters Condition: Required if TokenId is not specified.	Conditional
TokenId	The sender token ID that the Co-Branded service returned. Type: String Default: None Constraint: Max size = 65 characters Condition: Required if CallerReference is not specified.	Conditional

In your request, you must include parameters that are common to all requests. For more information, see “Common Request Parameters.”

Response Elements

Element	Description
Token	Details of the specified token. Type: Token

Responses also include elements common to all responses. For more information, see “Common Response Elements.”

Errors

This action can return the following errors:

- AccessFailure
- AccountClosed
- AuthFailure
- InternalError
- InvalidCallerReference
- InvalidClientTokenId
- InvalidParams
- InvalidTokenId
- SignatureDoesNotMatch

Sample REST Request

```
https://fps.sandbox.amazonaws.com?
Action=GetTokenByCaller
&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
&CallerReference=callerReferenceSingleUse10
&Signature=7E43HRAge3s57KDtEW3%2Fv0CE3Rh4TkVuOpk%2FIU%2FJIEY%3D
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2009-10-07T11%3A29%3A03.281Z
&TokenId=543IJMECGZZ3J4K1F7BJ3TMNXFBQU9VXNT7RRCTNAJDJ8X36L1ZRKSUU
PPIBTTIK
&Version=2008-09-17
```

Sample Response to REST Request

```
<GetTokenByCallerResponse
xmlns="http://fps.amazonaws.com/doc/2008-09-17/">
  <GetTokenByCallerResult>
    <Token>
      <TokenId>
543IJMECGZZ3J4K1F7BJ3TMNXFBQU9VXNT7RRCTNAJDJ8X36L1ZRKSUUPPIBTTIK
      </TokenId>
      <FriendlyName>Friendly1339359778</FriendlyName>
      <TokenStatus>Active</TokenStatus>
      <DateInstalled>2009-10-07T04:29:05.054-
07:00</DateInstalled>
      <CallerReference>callerReferenceSingleUse10</
CallerReference>
      <TokenType>SingleUse</TokenType>
      <OldTokenId>
543IJMECGZZ3J4K1F7BJ3TMNXFBQU9VXNT7RRCTNAJDJ8X36L1ZRKSUUPPIBTTIK
      </OldTokenId>
     <PaymentReason>PaymentReason</PaymentReason>
    </Token>
  </GetTokenByCallerResult>
  <ResponseMetadata>
    <RequestId>45b6c560-8aa9-463c-84be-80eeefb21034:0</RequestId>
  </ResponseMetadata>
</GetTokenByCallerResponse>
```

GetTransactionStatus

Description

The GetTransactionStatus action returns the status of the transaction specified by the TransactionId. You could use this action if you choose not to process Instant Payment Notifications (IPNs) that you receive from Amazon Payments (for more information, see “Setting Up Instant Payment Notification).”

Request Parameters

Parameter	Definition	Required
TransactionId	The transaction's ID. Type: String Constraint: Max size = 35 characters Default: None	Yes

In your request you must include parameters that are common to all requests. For more information, see “Common Request Parameters.”

Response Elements

Element	Description
CallerReference	A value you provide that uniquely identifies the request. Type: String Size: 128 bytes
StatusCode	Shorthand code that specifies the status of the transaction. Expands on the information in the TransactionStatus field. For example, if TransactionStatus is PENDING, this field might be PendingVerification, or PendingNetworkResponse. Type: String Size: 64 bytes Valid Values: See “Status Codes”
StatusMessage	A description of the transaction status. Type: String (LOB, Large Object)
TransactionId	Unique ID generated by Amazon FPS for this transaction. This element is returned if the transaction was accepted by Amazon FPS. Type: String

	Size: 35 Bytes
TransactionStatus	The status of the transaction. Provides a short code on the status of the transaction, for example "PENDING." Type: TransactionStatus Size: 20 bytes

Responses also include elements common to all responses. For more information, see “Common Response Elements.”

Status Codes

This action can return the following values for `StatusCode`.

Status Code	Message
Canceled	The transaction was explicitly canceled by the caller.
Expired	This reserved amount on the payment instrument was not settled within the timeout period OR The transaction could not be completed within the specified timeout.
PendingNetworkResponse	This transaction is awaiting a response from the backend payment processor OR (Message returned by backend payment processor)
PendingVerification	The transaction has been flagged for manual investigation
Success	The requested amount was reserved successfully against the given payment instrument. OR The transaction was successful and the payment instrument was charged.
TransactionDenied	(Message returned by backend payment processor). OR The transaction was denied after investigation.

Errors

This action can return the following synchronous errors, which occur within the status for this action.

- AccessFailure
- AuthFailure
- InternalError
- InvalidClientTokenId
- InvalidParams

- InvalidTransactionId
- SignatureDoesNotMatch

Sample REST Request

```
https://fps.sandbox.amazonaws.com?
Action=GetTransactionStatus
&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
&Signature=2l60qD6%2BDIfVEN7ZiHM0AcUKACZt0GYKFtIryqkCb6g%3D
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2009-10-06T09%3A12%3A06.921Z
&TransactionId=14GKE3B85HCMF1BTSH5C4PD2IHZL95RJ2LM
&Version=2008-09-17
```

Sample Query Request

```
GET\n
fps.sandbox.amazonaws.com\n
Action=GetTransactionStatus
&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
&Signature=2l60qD6%2BDIfVEN7ZiHM0AcUKACZt0GYKFtIryqkCb6g%3D
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2009-10-06T09%3A12%3A06.921Z
&TransactionId=14GKE3B85HCMF1BTSH5C4PD2IHZL95RJ2LM
&Version=2008-09-17
```

Sample Response to REST Request

```
<GetTransactionStatusResponse
xmlns="http://fps.amazonaws.com/doc/2008-09-17/">
  <GetTransactionStatusResult>
    <TransactionId>14GKE3B85HCMF1BTSH5C4PD2IHZL95RJ2LM</
TransactionId>
    <TransactionStatus>Success</TransactionStatus>
    <CallerReference>CallerReference07</CallerReference>
    <StatusCode>Success</StatusCode>
    <StatusMessage>
The transaction was successful and the payment instrument was
charged.
    </StatusMessage>
  </GetTransactionStatusResult>
  <ResponseMetadata>
    <RequestId>13279842-6f84-41ef-ae36-c1ededaf278d:0</RequestId>
  </ResponseMetadata>
</GetTransactionStatusResponse>
```

Pay

Description

The Pay action initiates a transaction to move funds from a sender to a recipient. The SenderTokenId, obtained from a Co-Branded service request, specifies the payment instrument the sender chose to execute the transaction. If the payment method specified is Amazon account balance transfer (ABT), the transaction completes synchronously. If the payment method is a bank account (ACH) or a credit card (CC), the transaction completes asynchronously.

The marketplace implementation of Pay includes the recipient token ID, which identifies the recipient. You get this in the response from a marketplace Co-Branded service request (which you make when the recipient signs up on your website for your marketplace services).The recipient token ID returned identifies the recipient and is required when you later move money from the sender to the recipient.

In addition, for marketplace applications, the Pay parameters also specify the marketplace fee and who is charged (the caller or recipient). The marketplace fee is typically the fee you charge the recipient for the service of hosting the recipient's e-commerce store. The fee can be charged on a per-transaction basis and consist of a flat fee, a percentage of the transaction, or a combination of the two.

Request Parameters

Parameter	Description	Required
CallerDescription	Description of this transaction for the caller. Type: String Default: None Constraint: Max size = 160 characters	No
CallerReference	A value you provide that uniquely identifies the request. For more information, see “Important Values to Store in Your Database.” Type: String Default: None Constraint: Max size = 128 characters	Yes
ChargeFeeTo	Specifies the participant paying the Amazon FPS fee in the transaction. The participant can only be a recipient or a caller. The following rules apply for specifying this parameter. • If you are playing the role of a	No

Parameter	Description	Required
	recipient and a caller, then set the value of this parameter to Recipient. • If you are playing the role of caller and facilitating the transaction between a sender and a recipient, where the recipient pays the fee, then the fee is collected from the funds that are received from the sender. • If you (caller) are paying the fees, then the fee is collected from your account balance. Ensure that you have a sufficient account balance to cover for the fees. If your account has an insufficient account balance, Amazon FPS rejects the transaction. • ChargeFeeTo must be set to be consistent with the value for the Recipient Token API recipientPaysFee parameter set when the recipient signed up for your marketplace services. Otherwise, you will get an error message. Type: String Default: Recipient Valid values: Recipient | Caller	
DescriptorPolicy	Specifies the entity whose name and contact details would be displayed in the sender's credit card or bank account statement. Type: DescriptorPolicy Default: None	No
MarketplaceFixedFee	Specifies the fee charged by the marketplace developer as a fixed amount of the transaction. The MarketplaceFixedFee is a separate fee from the Amazon Payments fee, which is paid by the caller or recipient. You can express the fixed fee as an amount, such as 10 to mean $10. If you charge a variable fee per transaction, use the MarketplaceVariableFee parameter. Important	No

Parameter	Description	Required
	The value for MarketPlaceFixedFee must be less than or equal to the amount specified for the recipient token. If not, an InvalidParams error is returned with the following messages: "The MarketPlaceFixedFee ($amount-specified) specified is greater than the maximum fixed fee ($amount-agreed) agreed by the recipient. Type: Amount Default: If both the MarketplaceFixedFee and the MarketplaceVariableFee are unspecified, then the corresponding maximum values, if any, from the recipient token are used.	
MarketplaceVariableFee	Specifies the fee charged by the marketplace developer as a percentage of the transaction. The MarketplaceVariableFee is a separate fee from the Amazon Payments fee and is paid by the recipient. You can express the variable fee as a decimal, such as 5 to mean 5%. If you charge a fixed amount per transaction, use the MarketplaceFixedFee parameter. Important The value for MarketPlaceVariableFee must be less than or equal to the amount specified for the recipient token. If not, an InvalidParams error is returned with the following messages: "The MarketPlaceVariableFee ($amount-specified) specified is greater than the maximum variable fee ($amount-agreed) agreed by the recipient. Type: Decimal Default: None	No
OverrideIPNURL	Specifies the URL to receive the Instant Payment Notification (IPN) for the request. The IPN URL set with this parameter takes precedence over the IPN URL set in your account. Type: String Default: None	No
RecipientTokenId	Specifies the recipient token used in the transaction. You obtain this value in response from the Co-Branded service	Conditional

Parameter	Description	Required
	Recipient Token API (for more information, see “Recipient Token API).” Type: String Default: None Condition: Required for marketplace transactions	
SenderDescription	Description of this transaction for the sender. If you use dynamic soft descriptors, you must specify a value for the sender description. Type: String Default: None Constraint: Max size = 160 characters Condition: If you use dynamic soft descriptors, you must specify a value for the sender description.	Conditional
SenderTokenId	Specifies the sender token used in the transaction. You obtain this value from the response to the Co-Branded service request. Type: String Default: None	Yes
TransactionAmount	Transaction amount charged to the sender for the purchase of an item or service. To understand how to correctly specify the amount in a REST request, see the example request at the end of this topic. Type: Amount Default: None	Yes
TransactionTimeoutInMins	Specifies the number of minutes before the request times out. Use this parameter to specify a timeout value that is acceptable for your business. If Amazon FPS cannot complete the transaction in the time allotted, the transaction is marked as failed and you receive an IPN notification (if you are using IPN). Type: Integer (number of minutes) Default: 10080 (seven days)	No

You must also include parameters that are common to all requests. For more information, see “Common Request Parameters.”

Response Elements

Element	Description

TransactionId	Unique ID generated by Amazon FPS for this transaction. This element is returned if the transaction was accepted by Amazon FPS. If the transaction is a Refund request, this parameter will contain the id of the Refund transaction only. Type: String Size: 35 Bytes
TransactionStatus	Provides the status of the transaction. Use this to determine if the transaction has completed, failed, or has not completed yet. Type: TransactionStatus.

Responses also include elements common to all responses. For more information, see “Common Response Elements.”

Pay careful attention to all of the response elements listed in the preceding table, especially the response status element which indicates success or failure for the Pay operation. If the response status is failure, the Errors element includes an error code that identifies the source of the failure. If the response status is success, the elements listed in the preceding table are returned.

Errors

This action can return the following errors:

- AccessFailure
- AccountLimitsExceeded
- AmountOutOfRange
- AuthFailure
- BadRule
- DuplicateRequest
- IncompatibleTokens
- InsufficientBalance
- InternalError
- InvalidAccountState_Caller
- InvalidAccountState_Recipient
- InvalidAccountState_Sender
- InvalidClientTokenId
- InvalidParams
- InvalidTokenId_Recipient
- InvalidTokenId_Sender
- NotMarketplaceApp
- PaymentMethodNotDefined
- SameSenderAndRecipient
- SameTokenIdUsedMultipleTimes
- SignatureDoesNotMatch
- TokenNotActive_Recipient
- TokenNotActive_Sender
- TokenUsageError
- TransactionDenied
- UnverifiedAccount_Recipient
- UnverifiedAccount_Sender
- UnverifiedBankAccount
- UnverifiedEmailAddress_Caller
- UnverifiedEmailAddress_Recipient
- UnverifiedEmailAddress_Sender

Sample REST Request

```
https://fps.sandbox.amazonaws.com?
Action=Pay
&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
&CallerDescription=MyWish
&CallerReference=CallerReference02
```

```
&SenderTokenId=553ILMLCG6Z8J431H7BX3UMN3FFQU8VSNTSRNCTAASDJNX66LN
ZLKSZU3PI7TXIH
&Signature=0AgvXMwJmLxwdMaiE7lMHZxc6384h%2FjBkiTserQFpBQ%3D
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2009-10-06T05%3A49%3A52.843Z
&TransactionAmount.CurrencyCode=USD
&TransactionAmount.Value=1
&Version=2008-09-17
```

Sample Response to REST Request

```
<PayResponse xmlns="http://fps.amazonaws.com/doc/2008-09-17/">
  <PayResult>
    <TransactionId>14GK6BGKA7U6OU6SUTNLBI5SBBV9PGDJ6UL</
TransactionId>
    <TransactionStatus>Pending</TransactionStatus>
  </PayResult>
  <ResponseMetadata>
    <RequestId>c21e7735-9c08-4cd8-99bf-535a848c79b4:0</RequestId>
  </ResponseMetadata>
</PayResponse>
```

Sample IPN Pending Notification to Rest Request

```
--------------------------
transactionId: 14GK6BGKA7U6OU6SUTNLBI5SBBV9PGDJ6UL
statusMessage: The transaction is awaiting a response from the
backend payment
processor.
transactionDate: 1254808208
signatureVersion: 2
signatureMethod: RSA-SHA1
buyerEmail: new_premium@amazon.com
notificationType: TransactionStatus
callerReference: CallerReference02
transactionAmount: USD 1.00
transactionStatus: PENDING
operation: PAY
recipientEmail: test-caller@amazon.com
buyerName: Test Business
signature:
uhP7uiCAvF/wTpRg6U279KTGPU2QHt23WiwNIB43i4ni1AEZOmBCTa3tUh1ugwxvI
MSRASBhiG0urUl22IAXbt1iXfYprM2VrS0W0/W23BpkxInuNeAQWKu4W5/uuOJ1gV
qyXsmxdFqJM7KKOh3IuUdCwSfvPooR2qDQ2r5H/HjcOHfWQZk+BknX1w+aYpBRTa/
mTYVxI6yq39mRyYPyMmh8r+tIPDevfnV1B7sRljhXkJZh6rHJEi7CHq4oqbf8HZ38
xaaqyggWy310SmMOuY3YcxNng0TOdbkgNAozMIQgfOsL4yxiyVIZZJEKFPgT/Odeb
CZkR/raY1JeuBdYOg==
recipientName: Test Business
```

```
paymentMethod: CC
certificateUrl:
https://fps.sandbox.amazonaws.com/certs/090909/PKICert.pem
paymentReason: MyWish
statusCode: PendingNetworkResponse
```

Sample IPN Success Notification to Rest Request

```
--------------------------
transactionId: 14GK6BGKA7U6OU6SUTNLBI5SBBV9PGDJ6UL
statusMessage: The transaction was successful and the payment
instrument was
charged.
transactionDate: 1254808208
signatureVersion: 2
signatureMethod: RSA-SHA1
buyerEmail: new_premium@amazon.com
notificationType: TransactionStatus
callerReference: CallerReference02
transactionAmount: USD 1.00
transactionStatus: SUCCESS
operation: PAY
recipientEmail: test-caller@amazon.com
buyerName: Test Business
signature:
yuYUR4IkONbOfrerafrzC6raA90suk+jKXCgaV1LY0DxieYCAG2tAf9S7Rt231kzr
0mhMMOIH0oeocHId3zdXp+2VaUbE4qGjPGfImpaBVxtxVwcdQP6cSFnvnKAbPbmQM
deIHMlgDeqVdtu5BO5skwje6bkDs+b8TQ3pHBYmXDc69aHceGqWAjMujs6m4HH3Ot
hlb5Rj54s1IedwTi63HyQo+IAyRWvGPTnnT6YlV0ajG38GCPoS9Wqa+UKcIr0sLoP
Y0y2StCDyjYHz7iVx+6lzG1eeCmZ++rAKU8swwhBiWGZ56ajlKTzhoIJnK5yk7jFY
reRt+Ff0W2fEnvEyQ==
recipientName: Test Business
paymentMethod: CC
certificateUrl:
https://fps.sandbox.amazonaws.com/certs/090909/PKICert.pem
paymentReason: MyWish
statusCode: Success
```

Related Actions

- Refund

Refund

Description

You use Refund to refund a successfully completed payment transaction. You can refund less than the amount paid. The default, however, is to refund the full amount to the sender.

Only the caller of the original transaction can perform a refund.

Request Parameters

Parameter	Description	Required
CallerDescription	Description of this transaction for the caller. Type: String Default: None Constraint: Max size = 160 characters	No
CallerReference	A value you provide that uniquely identifies the request. For more information, see “Important Values to Store in Your Database.” Type: String Default: None Constraint: Max size = 128 characters	Yes
OverrideIPNURL	Specifies the URL to receive the Instant Payment Notification (IPN) for the request. The IPN URL set with this parameter takes precedence over the IPN URL set in your account. Type: String Default: None	No
RefundAmount	Specifies the amount to be refunded. To understand how to correctly specify the amount in a REST request, see the example request at the end of this topic. Type: Amount Default: Original transaction amount or any amount remaining Constraint: The total refund amount cannot exceed the original transaction amount.	No
TransactionId	Transaction ID of the transaction to be refunded. Type: String Default: None Constraint: Max size = 35 characters	Yes

For your requests, you must include parameters that are common to all requests. For more information, see “Common Request Parameters.”

Response Elements

Element	Description
TransactionId	This is the ID (max size = 35 characters) of the transaction named in the request. Type: String Size: 35 Bytes
TransactionStatus	Provides the status of the transaction. Type: TransactionStatus

Responses also include elements common to all responses. For more information, see "Common Response Elements."

Errors

This action can return the following errors:

- AccessFailure
- AmountOutOfRange
- AuthFailure
- ConcurrentModification
- DuplicateRequest
- InternalError
- InvalidAccountState_Caller
- InvalidAccountState_Recipient
- InvalidAccountState_Sender
- InvalidClientTokenId
- InvalidParams
- InvalidTransactionId
- OriginalTransactionFailed
- OriginalTransactionIncomplete
- RefundAmountExceeded
- SignatureDoesNotMatch
- TransactionDenied
- TransactionFullyRefundedAlready
- TransactionTypeNotRefundable
- UnverifiedEmailAddress_Caller
- UnverifiedEmailAddress_Sender

Sample REST Request

```
https://fps.sandbox.amazonaws.com?
Action=Refund
&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
&CallerDescription=MyWish
&CallerReference=CallerReference03
&RefundAmount.CurrencyCode=USD
&RefundAmount.Value=1
&Signature=V6pU3PvDPkPhR9Eu7yZXnFZHuEFafLE5sBPgqqCELEU%3D
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2009-10-06T05%3A51%3A49.578Z
&TransactionId=14GK4TNCAQ84NK9VITEHKAS94RAD9ZE2AQD
&Version=2008-09-17
```

Sample Response to REST Request

```
<RefundResponse xmlns="http://fps.amazonaws.com/doc/2008-09-17/">
```

```
  <RefundResult>
    <TransactionId>
      14GK6F2QU755ODS27SGHEURLKPG72Z54KMF
    </TransactionId>
    <TransactionStatus>Pending</TransactionStatus>
  </RefundResult>
  <ResponseMetadata>
    <RequestId>1a146b9a-b37b-4f5f-bda6-012a5b9e45c3:0</RequestId>
  </ResponseMetadata>
</RefundResponse>
```

Sample IPN Pending Notification to REST Request

```
--------------------------
transactionId: 14GK6F2QU755ODS27SGHEURLKPG72Z54KMF
statusMessage: The transaction is awaiting a response from the
backend payment
processor.
transactionDate: 1254808324
signatureVersion: 2
signatureMethod: RSA-SHA1
parentTransactionId: 14GK4TNCAQ84NK9VITEHKAS94RAD9ZE2AQD
buyerEmail: new_premium@amazon.com
notificationType: TransactionStatus
callerReference: CallerReference03
transactionAmount: USD 1.00
transactionStatus: PENDING
operation: REFUND
recipientEmail: test-caller@amazon.com
buyerName: Test Business
signature:
mzis1HbeiiLx5j8nrUR3UeIVz3bcxVDG82JOW0gIEXO1FXxBVZHwPPBFCEVcyBMu8
wtNTMph/yluokjBi8w9Q6shMswBteq9bwNQA9qbDRT256ckoqdwfCf0910lYVj+wN
SKkezF6Clptjgsn0wMjMQOD9QBuOAAA9qV6VnUorRumPZ1psY/17FUvDwKVUMPEkZ
NO1mn7lcLFZJJp1aMkIj+RmraafTUUM62U0VMYKSR5pDEp0ifThn0Za4DogV0ZoGJ
rB/+gPhA07FdtnkM4uG5jgwqOCVyOA4ayP7uJpb7oImj8Jhi60+EWUUbbUShTEsjT
xqQtM8UKvsM6XAjdA==recipientName:
Test Business
paymentMethod: CC
certificateUrl:
https://fps.sandbox.amazonaws.com/certs/090909/PKICert.pem
paymentReason: MyWish
statusCode: PendingNetworkResponse
```

Sample IPN Success Notification to REST Request

```
--------------------------
transactionId: 14GK6F2QU755ODS27SGHEURLKPG72Z54KMF
```

```
statusMessage: The transaction was successful and the payment
instrument was charged.
transactionDate: 1254808324
signatureVersion: 2
signatureMethod: RSA-SHA1
parentTransactionId: 14GK4TNCAQ84NK9VITEHKAS94RAD9ZE2AQD
buyerEmail: new_premium@amazon.com
notificationType: TransactionStatus
callerReference: CallerReference03
transactionAmount: USD 1.00
transactionStatus: SUCCESS
operation: REFUND
recipientEmail: test-caller@amazon.com
buyerName: Test Business
signature:
sDq9YvW7L29W2NSIC/wjC5yLyR4QJSQyt/7iHhNiEdwFoGVkrLjJHiBloPfJxzznH
nmMtCRsUQ+Ad3tZ0NdemMxf0qYM9NX93PyG0KBKXShKeM0Da39cvnC05tZmtxpfCu
ZT5ECRydr+BqRo/DOlx1Yg93gihZ83qHWR8bpqQcBwsu7vD4c4m4mTZ4I75gw+NXK
RDD+vCPFDNEKRnh5kQz+Tjjg4bnNYEEcGRf6UZfS2lvMzdj0c37RUY6t4gQ3W3Z9G
/REGjC98JBuTimk/kc1HoSc+xe6WtAH/siNurisyqgoBHWnQM8iRqLEHj/m9y6vx5
EBHBokD1BJMIiiZNg==recipientName:
Test Business
paymentMethod: CC
certificateUrl:
https://fps.sandbox.amazonaws.com/certs/090909/PKICert.pem
paymentReason: MyWish
statusCode: Success
```

Related Actions

- Pay
- Reserve
- Settle

Reserve

Description

The Reserve operation reserves the total price of a purchase against the sender's payment instrument. To charge the payment instrument, you must subsequently issue a Settle request. A reserve authorization is only valid for 7 days. After that, Amazon FPS automatically cancels the transaction and notifies you.

> **Note**
> You can settle a reserved transaction only once. To cancel a reserved payment, send a Cancel request.

Request Parameters

Parameter	Description	Required
CallerDescription	Description of this transaction for the caller. Type: String Default: None Constraint: Max size = 160 characters Condition: If you use dynamic, soft descriptors, you must supply a caller description. For more information, see “DescriptorPolicy.”	No
CallerReference	A value you provide that uniquely identifies the request. For more information, see “Important Values to Store in Your Database.” Type: String Default: None Constraint: Max size = 128 characters	Yes
DescriptorPolicy	Specifies the entity whose name and contact details would be displayed in the sender's credit card or bank account statement. Type: Descriptor Policy Default: None	No
OverrideIPNURL	Specifies the URL to receive the Instant Payment Notification (IPN) for the request. The IPN URL set with this parameter takes precedence over the IPN URL set in your account. Type: String Default: None	No
SenderDescription	Description of this transaction for the sender. Type: String Default: None	Conditional

Parameter	Description	Required
	Constraint: Max size = 160 characters Condition: If you use dynamic soft descriptors, you must specify a value for the sender description. For more information, see “DescriptorPolicy.”	
SenderTokenId	Specifies the sender token to be used for this transaction. You obtain this value in a Co-Branded service response. Type: String Default: None	Yes
TransactionAmount	Transaction amount charged to the sender. To understand how to correctly specify the amount in a REST request, see the example request at the end of this topic. Type: Amount Default: None	Yes
TransactionTimeoutInMins	Specifies the number of minutes before the request times out. Use this parameter to specify a timeout value that is acceptable for your business. If Amazon FPS cannot complete the transaction in the time allotted, the transaction is marked as failed and you receive an IPN notification (if you are using IPN). Type: Integer (number of minutes) Default: 10080 (seven days)	No

For your requests you must include parameters that are common to all requests. For more information, see “Common Request Parameters.”

Response Elements

Element	Description
TransactionId	Unique ID generated by Amazon FPS for this transaction. This element is returned if the transaction was accepted by Amazon FPS. If the transaction is a Refund request, this parameter will contain the id of the Refund transaction only. Type: String Size: 35 Bytes
TransactionStatus	Provides the status of the transaction. Type: TransactionStatus.

Responses also include elements common to all responses. For more information, see “Common Response Elements.”

Errors

This action can return the following errors:

- AccessFailure
- AccountLimitsExceeded
- AmountOutOfRange
- AuthFailure
- DuplicateRequest
- IncompatibleTokens
- InternalError
- InvalidAccountState_Caller
- InvalidAccountState_Recipient
- InvalidAccountState_Sender
- InvalidClientTokenId
- InvalidParams
- InvalidPaymentMethod
- InvalidRecipientForCCTransaction
- InvalidTokenId_Sender
- PaymentInstrumentNotCC
- SignatureDoesNotMatch
- TokenNotActive_Recipient
- TokenNotActive_Sender
- TransactionDenied
- UnverifiedAccount_Recipient
- UnverifiedAccount_Sender
- UnverifiedEmailAddress_Caller
- UnverifiedEmailAddress_Recipient
- UnverifiedEmailAddress_Sender

Sample REST Request

```
https://fps.sandbox.amazonaws.com?
Action=Reserve
&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
&CallerDescription=Reserve
&CallerReference=CallerReference05
&SenderTokenId=553IPMACGAZ2J4N1L7BJ3UMNRFTQU4V9NT4RJCTVADDJKXQ6L1
ZAKSIUNPIRTTI1
&Signature=JZ0eeVTM5LwbvziLdA%2FSMve7mgrEoTvTGZJ%2BpsgZkM0%3D
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2009-10-06T07%3A51%3A04.140Z
&TransactionAmount.CurrencyCode=USD
&TransactionAmount.Value=1
&Version=2008-09-17
```

Sample Response to REST Request

```
<ReserveResponse xmlns="http://fps.amazonaws.com/doc/2008-09-
17/">
  <ReserveResult>
    <TransactionId>14GKD9GE66FAA63E6O6B2JDPZKN53LZ7F22</
TransactionId>
    <TransactionStatus>Pending</TransactionStatus>
  </ReserveResult>
  <ResponseMetadata>
    <RequestId>d13273fc-fca8-4963-8fbc-66d03e66055f:0</RequestId>
  </ResponseMetadata>
</ReserveResponse>
```

Sample IPN Pending Notification to REST Request

```
--------------------------
transactionId: 14GKD9GE66FAA63E6O6B2JDPZKN53LZ7F22
statusMessage: The transaction is awaiting a response from the
backend payment
processor.
transactionDate: 1254815482
signatureVersion: 2
signatureMethod: RSA-SHA1
buyerEmail: new_premium@amazon.com
notificationType: TransactionStatus
callerReference: CallerReference05
operation: RESERVE
transactionStatus: PENDING
transactionAmount: USD 1.00
recipientEmail: test-caller@amazon.com
buyerName: Test Business
signature:
NvFCZMralNEepynuIhhXJc+jpK1ZMdFLBMcXFv6Vq1jhpdLX/B9T0lluOUv74I6xg
O8L2UemgV4SZCejlQZ3glwKnEM75lKVlHx34IKp1RFm1DjQOO5KaYGQUNMu1ouYK1
YmQUHCuktdLnTXjkxjn0lv9U4EyzDe8l/tLp2nlAqRF4J7PIhdTkWvBYNYhZrEy5A
895OMf9uFtwX8Eyg4lTDMVwEWJoG8CTxJqtcsKabmbF9Blwhfe3f+viTnv39YRDb+
PZKnpl/XqkKYdNEXClRy3g6xpF/14FJ4hA+A1UP+A+No17b6lZuKmd5dbdvqTQKOx
EAfR6lL1gTzAYY/8w==
recipientName: Test Business
paymentMethod: CC
certificateUrl:
https://fps.sandbox.amazonaws.com/certs/090909/PKICert.pem
paymentReason: Reserve
statusCode: PendingNetworkResponse
```

Sample IPN Success Notification to REST Request

```
--------------------------
transactionId: 14GKD9GE66FAA63E6O6B2JDPZKN53LZ7F22
statusMessage: The requested amount was reserved successfully
against the given
payment instrument.
transactionDate: 1254815482
signatureVersion: 2
signatureMethod: RSA-SHA1
buyerEmail: new_premium@amazon.com
notificationType: TransactionStatus
callerReference: CallerReference05
transactionAmount: USD 1.00
transactionStatus: RESERVED
operation: RESERVE
recipientEmail: test-caller@amazon.com
buyerName: Test Business
```

```
signature:RIVZQHF+NmGUEbZNXijRcSwmeBTcYg/GCZD/xeUpLLXMwDNrM1D0+ew
FLiUqJvdbQueUilBkJPoB5j+ZYvvrXfldEofaMZ85pz2pA/DyUicWR4e/DgcZrk/B
7FO6LL9ki6aE0qPzpRR/nzRcLiu1lH2azUPnMVf3dT+SfDhaKyKIfX40QYL6U3m3N
TaGYSUbBwzZczg9qTpu4zZ2kCK3uidg7P78sXQEnDhm8kDAJC4obYFVlZi/Bd8Ual
xIYf2ko8SkhQ4vbsipjNg++HJ7KlJAa41GTVCrJfeX0Y4r7ToONEaQiu/zn8X+q/j
PqgGZN+Z2KNls6XVw4Waw3eXbug==
recipientName: Test Business
paymentMethod: CC
certificateUrl:
https://fps.sandbox.amazonaws.com/certs/090909/PKICert.pem
paymentReason: Reserve
statusCode: Success
```

Related Actions

- Pay
- Refund
- Settle

Settle

Description

The Settle action charges the sender's payment instrument for the purchase that was transacted using Reserve. You settle a transaction when you fulfill the order, for example, when you ship the purchased items.

Request Parameters

Parameter	Description	Required
OverrideIPNURL	Specifies the URL to receive the Instant Payment Notification (IPN) for the request. The IPN URL set with this parameter takes precedence over the IPN URL set in your account. Type: String Default: None	No
ReserveTransactionId	An identifier returned by Reserve that identifies the reserved transaction to be settled. Type: String Default: None Constraint: Max size = 35 Characters	Yes
TransactionAmount	Amount to be settled. To understand how to correctly specify the amount in a REST request, see the example request at the end of this topic. Type: Amount Default: The amount reserved in the Reserve request Constraint: The amount cannot exceed the reserved amount.	No

For your requests you must include parameters that are common to all requests. For more information, see “Common Request Parameters.”

Response Elements

Element	Description
TransactionId	Identifies the transaction that was settled. Type: String Size: 35 Bytes
TransactionStatus	Provides the status of the transaction. Type: TransactionStatus

Responses also include elements common to all responses. For more information, see “Common Response Elements.”

Errors

This action can return the following errors:

- AccessFailure
- AccountClosed
- AmountOutOfRange
- AuthFailure
- ConcurrentModification
- InternalError
- InvalidAccountState_Caller
- InvalidAccountState_Recipient
- InvalidAccountState_Sender
- InvalidClientTokenId
- InvalidParams
- InvalidTransactionId
- InvalidTransactionState
- SettleAmountGreaterThanReserveAmount
- SignatureDoesNotMatch
- TransactionDenied
- UnverifiedAccount_Recipient
- UnverifiedEmailAddress_Caller
- UnverifiedEmailAddress_Recipient
- UnverifiedEmailAddress_Sender

Sample REST Request

```
https://fps.sandbox.amazonaws.com?
Action=Settle
&AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
&ReserveTransactionId=14GKD9GE66FAA63E6O6B2JDPZKN53LZ7F22
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Signature=SJJLsIBghi7VIycBjX7c3hnfgZ%2FBvZbzqLtAZXDL8ys%3D
&Timestamp=2009-10-06T07%3A53%3A11.750Z
&TransactionAmount.CurrencyCode=USD
&TransactionAmount.Value=1
&Version=2008-09-17
signature value>
```

Sample Response to REST Request

```
<SettleResponse xmlns="http://fps.amazonaws.com/doc/2008-09-17/">
  <SettleResult>
    <TransactionId>14GKD9GE66FAA63E6O6B2JDPZKN53LZ7F22</
TransactionId>
    <TransactionStatus>Pending</TransactionStatus>
  </SettleResult>
  <ResponseMetadata>
    <RequestId>9ed2008b-b230-4ed0-9210-095f77fc2359:0</RequestId>
  </ResponseMetadata>
</SettleResponse>
```

Sample IPN Pending Notification to Rest Request

```
--------------------------
transactionId: 14GKD9GE66FAA63E6O6B2JDPZKN53LZ7F22
statusMessage: The transaction is awaiting a response from the
backend payment
processor.
transactionDate: 1254815482
signatureVersion: 2
signatureMethod: RSA-SHA1
buyerEmail: new_premium@amazon.com
notificationType: TransactionStatus
callerReference: CallerReference05
transactionAmount: USD 1.00
transactionStatus: PENDING
operation: SETTLE
recipientEmail: test-caller@amazon.com
buyerName: Test Business
signature: zxymWMlhu4o+2rp
drBXu08EACZ3Mi3Z16x5+8+1Hbqkh4DTr1A6ry4fijBYkl32z4fMF9xnoGriW2jzi
j7Vmc/4Vc4dEWCpbOq+be4JLfOELw08jJQintuk3kIXOPca06NMWQhGiC3m7kRF95
nM2TJs7jqbkAMrKyiZArcURMo0YpRZPIF7DlDlNRAebH2+0v0BxaUtombrDFW4UlS
scuebXDNdgjp7KjCnTBJGDJks9/wLKKvFtISQWHuvN2MiPzt7UmFwMLPh8jtpgQ6J
xS+ipTPxbr7Km3IXIJJgJHpxmdQmgghrl4IX0zCKaVUb7Rh3z85/9F0yPB8A92nqu
zQ==
recipientName: Test Business
paymentMethod: CC
certificateUrl:
https://fps.sandbox.amazonaws.com/certs/090909/PKICert.pem
paymentReason: Reserve
statusCode: PendingNetworkResponse
```

Sample IPN Success Notification to REST Request

```
--------------------------
transactionId: 14GKD9GE66FAA63E6O6B2JDPZKN53LZ7F22
statusMessage: The transaction was successful and the payment
instrument was
charged.
transactionDate: 1254815482
signatureVersion: 2
signatureMethod: RSA-SHA1
buyerEmail: new_premium@amazon.com
notificationType: TransactionStatus
callerReference: CallerReference05
operation: SETTLE
transactionStatus: SUCCESS
transactionAmount: USD 1.00
recipientEmail: test-caller@amazon.com
buyerName: Test Business
```

```
signature:
pwozZP+lYONFq39g13ux44vFFMRAt4eJ9kOUWMV2uPCrvBqzi4LFYDQY5UE3VW8OU
iW+qpbukqFzYNvE+8mh7adhX/qee2U8ZUUNZi6LaM3sKtpPxus2ZJ3wDVPjuO02Ob
tu1G6Eo79iMi8viX7Dz1LL8pFTdhspHZb0XDWkuOt2pK2aELa7TOZ/pXXUFLvGrn4
MOd6INwbyM2fvnJpIDTcNdzedBO3Rw3vp2f2GfpFAZJD6Imu57rsr9RsHVUqu2bIh
JaAgTRFleVKzMHQJqft5jo6M9N4vKmPfccsuAvoF+rDn+/6a9VEvTBrVcvAhJ5jrB
p3FkXYkOPbHchqHfQ==
recipientName: Test
BusinesspaymentMethod: CC
certificateUrl:
https://fps.sandbox.amazonaws.com/certs/090909/PKICert.pem
paymentReason: Reserve
statusCode: Success
```

Related Actions

- Pay
- Refund
- Reserve

VerifySignature

Description

VerifySignature enables you to verify the signature included with outbound notifications. A correctly formatted call using VerifySignature returns a positive result when the signature is valid for the response that contained it.

This action is a component of signature version 2. Because of this, you may only use it with responses which have a SignatureVersion value of 2. As of 10 February, 2011, Amazon Payments signs all outbound responses with signature version 2. Unsigned outbound responses are no longer supported.

Note
You sign VerifySignature as you would any other Amazon FPS action.

Request Parameters

Parameter	Description	Required
UrlEndPoint	A required field that contains the appropriate originating endpoint (either the returnUrl or ipnUrl) that received the response. For example, if your web application resides at http://my-app-website.biz/, the returnUrl might be http://my-app-website.biz/amazon/success.php, and the IPNUrl might be http://my-app-website.biz/amazon/ipnProcessor.php. Type: String	Yes

Parameter	Description	Required
	Default: None Constraint: Cannot be null or empty	

HttpParameters	Concatenated string of all URL-Encoded parameters which were included in the response containing the signature you want to verify. This includes the certificateUrl, signatureVersion, signatureMethod and signature parameters. For example, a correctly formatted and URL-encoded string resembles the following:
	`First%20Name=Joe&Last%20Name=Smith&signa`
	`tureVersion=2&signatureMethod=HMACSHA256`
	`&certificateUrl=https%253A%252F%252Ffps.`
	`amazon`
	`aws.com%252Fcert%252Fkey.pem&signature=a`
	`oeuAOE123eAUdhf]`
	Tip
	For validating the returnUrl, you can extract the query string from the returnUrl (excluding the '?' character). For validating the IPNUrl, concatenate the POST parameters.
	Type: String
	Default: None
	Constraint: Cannot be null or empty. In addition, because VerifySignature is a component of signature version 2, the value for signatureVersion must be 2.
		Yes

You must also use the Action parameter as described in Common Request Parameters. Parameter names are case sensitive.

Response Elements

Element	Description
VerificationStatus	The result of the verification, either Success or Failure. Type: VerificationStatus

Responses also include elements common to all responses. For more information, see “Common Response Elements.”

Errors

This action can return the following errors:

- InternalServerError
- InvalidParam

Sample REST Request

```
https://fps.sandbox.amazonaws.com/?Action=VerifySignature&UrlEndP
oint=http%3A%2F%2Fexample.com%3A8080%2Fipn.jsp&HttpParameters=exp
iry%3D08%252F2015%26signature%3DynDukZ9%252FG77uSJVb5YM0cadwHVwYK
PMKOO3PNvgADbv6VtymgBxeOWEhED6KGHsGSvSJnMWDN%252FZl639AkRe9Ry%252
F7zmn9CmiM%252FZkp1XtshERGTqi2YL10GwQpaH17MQqOX3u1cW4LlyFoLy4celU
FBPq1WM2ZJnaNZRJIEY%252FvpeVnCVK8VIPdY3HMxPAkNi5zeF2BbqH%252BL2vA
Wef6vfHkNcJPlOuOl6jP4E%252B58F24ni%252B9ek%252FQH18O4kw%252FUJ7Zf
KwjCCI13%252BcFybpofcKqddq8CuUJj5Ii7Pdw1fje7ktzHeeNhF0r9siWcYmd4J
axTP3NmLJdHFRq2T%252FgsF3vK9m3gw%253D%253D%26signatureVersion%3D2
%26signatureMethod%3DRSA-
SHA1%26certificateUrl%3Dhttps%253A%252F%252Ffps.sandbox.amazonaws
.com%252Fcerts%252F090909%252FPKICert.pem%26tokenID%3DA5BB3HUNAZF
J5CRXIPH72LIODZUNAUZIVP7UB74QNFQDSQ9MN4HPIKISQZWPLJXF%26status%3D
SC%26callerReference%3DcallerReferenceMultiUse1&AWSAccessKeyId=AK
IAIOSFODNN7EXAMPLE&Timestamp=2010-02-
26T19%3A48%3A05.000Z&Version=2008-09-
17&SignatureVersion=2&SignatureMethod=HmacSHA256&Signature=fKRGL4
2K7nduDA47g6bJCyUyF5ZvkBotXE5jVcgyHvE%3D
```

Sample Query Request

```
GET\n
fps.sandbox.amazonaws.com\nAction=VerifySignature&UrlEndPoint=htt
p%3A%2F%2Fexample.com%3A8080%2Fipn.jsp&HttpParameters=expiry%3D08
%252F2015%26signature%3DynDukZ9%252FG77uSJVb5YM0cadwHVwYKPMKOO3PN
vgADbv6VtymgBxeOWEhED6KGHsGSvSJnMWDN%252FZl639AkRe9Ry%252F7zmn9Cm
iM%252FZkp1XtshERGTqi2YL10GwQpaH17MQqOX3u1cW4LlyFoLy4celUFBPq1WM2
ZJnaNZRJIEY%252FvpeVnCVK8VIPdY3HMxPAkNi5zeF2BbqH%252BL2vAWef6vfHk
NcJPlOuOl6jP4E%252B58F24ni%252B9ek%252FQH18O4kw%252FUJ7ZfKwjCCI13
%252BcFybpofcKqddq8CuUJj5Ii7Pdw1fje7ktzHeeNhF0r9siWcYmd4JaxTP3NmL
JdHFRq2T%252FgsF3vK9m3gw%253D%253D%26signatureVersion%3D2%26signa
tureMethod%3DRSA-
SHA1%26certificateUrl%3Dhttps%253A%252F%252Ffps.sandbox.amazonaws
.com%252Fcerts%252F090909%252FPKICert.pem%26tokenID%3DA5BB3HUNAZF
J5CRXIPH72LIODZUNAUZIVP7UB74QNFQDSQ9MN4HPIKISQZWPLJXF%26status%3D
SC%26callerReference%3DcallerReferenceMultiUse1&AWSAccessKeyId=AK
IAIOSFODNN7EXAMPLE&Timestamp=2010-02-
26T19%3A48%3A05.000Z&Version=2008-09-
17&SignatureVersion=2&SignatureMethod=HmacSHA256&Signature=fKRGL4
2K7nduDA47g6bJCyUyF5ZvkBotXE5jVcgyHvE%3D
```

Sample Response to REST Request

```
<VerifySignatureResponse
xmlns="http://fps.amazonaws.com/doc/2008-09-17/">
  <VerifySignatureResult>
    <VerificationStatus>Success</VerificationStatus>
  </VerifySignatureResult>
  <ResponseMetadata>
    <RequestId>197e2085-1ed7-47a2-93d8-d76b452acc74:0</RequestId>
  </ResponseMetadata>
</VerifySignatureResponse>
```

Common Request Parameters

Each action in the API has its own specific set of parameters, but there is also a set of parameters that all actions use. This section describes those input parameters.

The following table describes parameters that can be used in all requests.

Parameter	Description	Required
Action	The API operation, for example, Settle or Refund. Type: String: Default: None Constraint: Must be a valid operation such as Cancel, Refund, and so on.	Yes
AWSAccessKeyId	A string, distributed by Amazon FPS when you sign up to be a developer that uniquely identifies the caller. Type: String Default: None	Yes
Signature	A value calculated using the request parameters and a SHA256 (preferred) or SHA1 HMAC encryption algorithm. Type: String Default: None	Yes
SignatureVersion	A value that specifies the Signature format. Type: Integer Default: None Valid Value: 2 **Important** The previous method for signing (signature version 1) was deprecated on November 3rd, 2009, and as of 10 February, 2011 it is no longer supported. Whenever you sign a request with your access keys, you must now use signature version 2.	Yes
SignatureMethod	A value that specifies the signing method. Type: String Default: None Valid Values: HmacSHA256 (preferred) and HmacSHA1	Yes
Timestamp	A date-time value that marks the day and time the request was sent. Requests expire after a certain length of time to prevent malicious users from capturing requests and resubmitting them at a later time. Type: dateTime, for example, 2008-09-18T13:00:01Z Default: None	Yes
Version	The version number of the WSDL to use in processing the	Yes

Parameter	Description	Required
	request. Version numbers are dates, such as 2008-09-17. For a list of version numbers, go to the Amazon Resource Center at http://aws.amazon.com/resources. Type: String Default: None	

Common Response Elements

Each action in the API has its own set of response elements it uses. There are, however, a set of response elements that all actions use. The following table describes those common elements.

Element	Description
ResponseMetadata	Container element.
RequestId	Amazon FPS returns a RequestId element for every API call accepted for processing. The request ID is a reference to your API request that Amazon FPS can use to troubleshoot any issues related to the request. We recommend you store the request ID value for future reference. Because responses and requests can return asynchronously, you can use the request ID to sync responses with requests. Type: String Max Size: 64 Bytes
signatureVersion	A value that specifies the Signature format. Type: Integer Valid Values: 2
signatureMethod	A value that specifies the signing method. Type: String Valid Values: HmacSHA256 (preferred) and HmacSHA1.

Errors

Error	Description
AccessFailure	Account cannot be accessed. You can display the following message to your customers: Your account cannot be accessed. Retriable: Yes
AccountClosed	Account is not active. You can display the following message to your customers: Your account is closed.

Error	Description
	Retriable: Yes
AccountLimitsExceeded	The spending or receiving limit on the account is exceeded. This error can also occur when the specified bank account has not yet been verified. You can display the following message to your customers: You have exceeded your spending or receiving limits. You can view your current limits at http://payments.amazon.com/sdui/sdui/viewlimits. You can upgrade these limits by adding and verifying a bank account as a payment method. Please visit Adding and Verifying Bank Accounts to learn how to add and instantly verify a bank account. Retriable: Yes
AmountOutOfRange	The transaction amount is more than the allowed range. Ensure that you pass an amount within the allowed range. The transaction amount in a Pay operation using credit card or bank account must be greater than $0.01. Retriable: No
AuthFailure	AWS was not able to validate the provided access credentials. Please make sure that your AWS developer account is signed up for FPS. Retriable: Yes
ConcurrentModification	A retriable error can happen when two processes try to modify the same data at the same time. The developer should retry the request if this error is encountered. Retriable: Yes
DuplicateRequest	A different request associated with this caller reference already exists. You have used the same caller reference in an earlier request. Ensure that you use unique caller references for every new request. Even if your earlier request resulted in an error, you should still use a unique caller reference with every request and avoid this error.

Error	Description
	Retriable: No
InactiveInstrument	Payment instrument is inactive. The payment instrument is inactive, for example, a credit card has expired. Retriable: No
IncompatibleTokens	The transaction could not be completed because the tokens have incompatible payment instructions. If any assertion in one of the payment instructions fails, this error is displayed. As such, it may be caused by a number of reasons, for example: One or more tokens has expired. The recipient specified in the token is different from the actual recipient in the transaction. There is violation on the amount restriction. This token cannot be used with your application as another application has installed it.
InsufficientBalance	The sender, caller, or recipient’s account balance has insufficient funds to complete the transaction. You must ask your customers to fund their accounts. You can then retry this request. Funding an account can take up to three to four business days using a bank account transfer. This error is also displayed if the party paying the Amazon FPS fees does not have a sufficient account balance. Retriable: Yes
InternalError	A retriable error that happens due to some transient problem in the system. The caller should retry the API call if this error is encountered. Retriable: Yes
InvalidAccountState	The account is either suspended or closed. Payment instructions cannot be installed on this account. You must ask your customer to set up a new account if the account is closed. Retriable: Yes

Error	Description
InvalidAccountState_Caller	The developer account cannot participate in the transaction. Your account is not active. Contact your AWS Representative for more information. Retriable: Yes
InvalidAccountState_Recipient	Recipient account cannot participate in the transaction. You can display the following message to your customer (sender): Your Amazon Payments account is not active. Please visit http://payments.amazon.com for more details. Retriable: Yes
InvalidAccountState_Sender	Sender account cannot participate in the transaction. You can display the following message to your customer (sender): Your Amazon Payments account is not active. Please visit http://payments.amazon.com for more details. Retriable: Yes
InvalidCallerReference	The Caller Reference does not have a token associated with it. Use the caller reference value that was passed to the InstallPaymentInstruction operation or the Amazon FPS Co-Branded UI pipeline.
InvalidClientTokenId	The AWS Access Key Id you provided does not exist in our records. Please check that the AWS Access Key Id used to make the request is valid. Retriable: No
InvalidDateRange	The end date specified is before the start date or the start date is in the future. Specify the correct end date.
InvalidParams	One or more parameters in the request is invalid. For more information, see the parameter descriptions for the action in the API Reference. Parameters are case sensitive. Retriable: No

Error	Description
InvalidPaymentInstrument	The payment method used in the transaction is invalid. Specify a valid payment method
InvalidPaymentMethod	The cause for this error is dependent on the calling action: For InstallPaymentInstruction, payment method specified in the GK construct is invalid. Specify the correct payment method.
InvalidRecipientForCCTransaction	This account cannot receive credit card payments. You can display the following message to your customers: You cannot receive credit card payment. Please visit http://payments.amazon.com to update your account to receive credit card payments."
InvalidSenderRoleFor AccountType	This token cannot be used for this operation. Ensure that the account used in this transaction is the same account used in the original transaction. In a refund transaction, the recipient making the refund payment must be the same recipient as in the original transaction. Retriable: No
InvalidTokenId	You did not install the token that you are trying to cancel. You do not have permission to cancel this token. You can cancel only the tokens that you own. Retriable: No
InvalidTokenId_Recipient	You did not install the token that you are trying to cancel. You do not have permission to cancel this token. You can cancel only the tokens that you own. Retriable: No
InvalidTokenId_Sender	The send token specified is either invalid or canceled or the token is not active. You must ask your customer to set up a new payment authorization. Retriable: No
InvalidTokenType	An invalid operation was performed on the token, for example, getting the token usage information on a single use token.

Error	Description
	Retriable: No
InvalidTransactionId	The specified transaction could not be found or the caller did not execute the transaction or this is not a Pay or Reserve call. Specify the correct the transaction ID. Retriable: No
InvalidTransactionState	The transaction is not complete, or it has temporarily failed. Specify a duration of more than one hour. Retriable: No
NotMarketplaceApp	This is not an marketplace application or the caller does not match either the sender or the recipient. Please check that you are specifying the correct tokens. Retriable: Yes
OriginalTransactionFailed	The original transaction has failed. You cannot refund a transaction that has originally failed. Retriable: No
OriginalTransactionIncomplete	The original transaction is still in progress. Retry after the original transaction has completed. Retriable: Yes
PaymentInstrumentNotCC	The payment method specified in the transaction is not a credit card. You can only use a credit card for this transaction. Use only a credit card for this transaction.
PaymentMethodNotDefined	Payment method is not defined in the transaction. Specify the payment method in the sender token.
RefundAmountExceeded	The refund amount is more than the refundable amount. You are not allowed to refund more than the original transaction amount. Retriable: No
SameSenderAndRecipient	The sender and receiver are identical, which is not allowed.

Error	Description
	Retriable: No
SameTokenIdUsedMultipleTimes	This token is already used in earlier transactions. The tokens used in a transaction should be unique.
SenderNotOriginalRecipient	The sender in the refund transaction is not the recipient of the original transaction. The token you passed as the refund sender token does not belong to the recipient of the original transaction. Pass the correct refund sender token. Retriable: No
SettleAmountGreaterThanDebt	The amount being settled or written off is greater than the current debt. You cannot settle an amount greater than what is owed. Retriable: No
SettleAmountGreaterThan ReserveAmount	The amount being settled is greater than the reserved amount. You cannot settle an amount greater than what is reserved. Retriable: No
SignatureDoesNotMatch	The request signature calculated by Amazon does not match the signature you provided. Check your AWS Secret Access Key and signing method. For more information, see "Working with Signatures" in the Amazon Flexible Payments Service Getting Started Guide. Retriable: No
TokenAccessDenied	Permission is denied to cancel the token. You are not allowed to cancel this token. Retriable: No
TokenNotActive	The token is canceled. A new token needs to be created. Retriable: No
TokenNotActive_Recipient	The recipient token is canceled. If you are the recipient, set up a new recipient token using the InstallPaymentInstruction

Error	Description
	operation or direct your customers to the Recipient Token Installation Pipeline to set up recipient token. Retriable: No
TokenNotActive_Sender	The sender token is canceled. You must ask your customer to set up a new payment authorization because the current authorization is not active. Retriable: No
TokenUsageError	The token usage limit is exceeded. If the usage has exceeded for this period, then wait for the next period before making another transaction. If the usage has exceeded for the entire authorization period, then ask your customer to set up a new payment authorization.
TransactionDenied	This transaction is not allowed. You are not allowed to do this transaction. Check your credentials. Retriable: No
TransactionFullyRefunded Already	This transaction has already been completely refunded. You are not allowed to refund more than the original transaction amount. Retriable: No
TransactionTypeNotRefundable	You cannot refund this transaction. Refund is allowed only on the Pay operation. Retriable: No
UnverifiedAccount_Recipient	The recipient's account must have a verified bank account or a credit card before this transaction can be initiated. You can display the following message to your customer (recipient): Your Amazon Payments account is not active. Please visit http://payments.amazon.com for more details. Retriable: No
UnverifiedAccount_Sender	The sender's account must have a verified U.S.

Error	Description
	credit card or a verified U.S bank account before this transaction can be initiated. You can display the following message to your customers: Please add a U.S. credit card or U.S. bank account and verify your bank account before making this payment. Retriable: No
UnverifiedBankAccount	A verified bank account should be used for this transaction. Visit the http://payments.amazon.com web site to verify your bank account. Retriable: No
UnverifiedEmailAddress_Caller	The caller account must have a verified email address. You cannot make a web service API call without verifying your email address. Go to http://payments.amazon.com web site and make payments. Retriable: No
UnverifiedEmailAddress_ Recipient	The recipient account must have a verified email address for receiving payments. You can display the following message to your customers: You cannot receive payments. Please verify your email address. Go to http://payments.amazon.com to verify your account and receive payments. Retriable: No
UnverifiedEmailAddress_Sender	The sender account must have a verified email address for this payment You can display the following message to your customers: You cannot make payments. Please verify your email address. Go to http://payments.amazon.com to verify your account and make payments. Retriable: No

Data Types

This section describes the data types common to the Amazon FPS actions.

- Enumerated Data Types
- Complex Data Types

Enumerated Data Types

This section describes the enumerated data types Amazon FPS uses.

AccountBalance

Name	Description	Type
AvailableBalances	The total amount of money that is transferred to your account from a bank account transfer or a refund.	AvailableBalances
PendingInBalance	The total amount that is yet to be credited to your account.	Amount
PendingOutBalance	The total amount that is yet to be debited from your account.	Amount
TotalBalance	The total balance that is currently available in your account.	Amount

ChargeFeeTo

Name	Description	Type
Caller	Caller shall pay the fees.	String
Recipient	Recipient shall pay the fees	String

CurrencyCode

Name	Description	Type
USD	The transaction uses U.S. dollars.	String

FPSOperation

These values are returned for non-IPN operations.

Name	Description	Type
Pay	All pay transactions.	String
Refund	All refund transactions.	String
Settle	All settle transactions.	String
Reserve	All reserve transactions.	String

These values are returned only for IPN operations.

Name	Description	Type
PAY	All pay transactions.	String
REFUND	All refund transactions.	String
SETTLE	All settle transactions.	String
RESERVE	All reserve transactions.	String
MULTI_SETTLE	All multi-settle transactions.	String
REAUTH	All transactions that required reauthorization.	String
DEPOSIT_FUNDS	All fund deposit transactions.	String
WITHDRAW_FUNDS	All fund withdrawal transactions.	String
CANCEL_TRANSACTION	All non user cancelled transactions.	String
CANCEL	All user cancelled transactions. String	String

InstrumentId

Name	Description	Type
InstrumentId	An alphanumeric value that represents the payment instrument.	String Max size = 64 characters

InstrumentStatus

Name	Description	Type
Active	All active instruments installed String for your application	String
All	All instruments installed for your String application.	String
Cancelled	All canceled instruments.	String

PaymentMethod

Name	Description	Type
ABT	Amazon Payments account balance transfer	String
ACH	Bank account transaction.	String
CC	Credit card transaction.	String

RelationType

Name	Description	Type
MarketplaceFee	Marketplace fee transactions.	String
Parent	Parent transactions.	String
Refund	Refund transactions.	String
RefundReversal	RefundReversal transactions.	String
Reserve	Reserve transactions.	String
Settle	Settle transactions.	String

SortOrderByDate

Name	Description	Type
Ascending	Return results in ascending order by date.	String
Descending	Return results in descending order by date (default).	String

TokenStatus

Name	Description	Type
Active	The token is in active state.	String
Inactive	The token was canceled by the user and is inactive.	String

TokenType

Name	Description	Type
MultiUse	Token that can be used multiple times.	String
Recurring	Token which is specifically marked for recurring payments.	String
SingleUse	Token that can be used only once.	String
Unrestricted	Token with unrestricted usage. Sender tokens with unlimited usage cannot be installed by external applications. Only recipient tokens can be installed with unrestricted usage.	String

TransactionalRole

Name	Description	Type
Caller	Role is the caller.	String
Recipient	Role is the recipient.	String
Sender	Role is the sender.	String

TransactionStatus

These values are returned for non-IPN operations.

Name	Description	Type
Cancelled	The transaction was canceled.	String
Failure	The transaction failed. The API operation failed and Amazon FPS did not receive or record a transaction. You can retry the transaction only if a retriable error was returned.	String
Pending	The transaction is pending.	String
Reserved	The reserve request on the transaction succeeded. Amazon FPS reserves the purchase price against the sender's payment instrument.	String
Success	The transaction succeeded. You can fulfill the order for the customer.	String

TransactionStatus (IPN)

These values are returned for IPN operations only.

Name	Description	Type
CANCELLED	The transaction was canceled.	String
FAILURE	The transaction failed. The API operation failed and Amazon FPS did not receive or record a transaction. You can retry the transaction only if a retriable error has been returned.	String
PENDING	The transaction is pending.	String
RESERVED	The reserve request on the transaction succeeded. Amazon FPS reserves the purchase price against the sender's payment instrument.	String
SUCCESS	The transaction succeeded. You can fulfill the order for the customer.	String

Complex Data Types

This section describes the complex data types Amazon FPS uses.

Amount

Name	Description	Type
CurrencyCode	The currency code of the amount. Amazon FPS currently supports only USD.	CurrencyCode
Value	The numeric value of the amount in dollars. Two optional decimal places are allowed. For example, 25.01 is $25.01, and 2500 is $2500.	String

AvailableBalances

Name	Description	Type
DisburseBalance	The total balance that has been disbursed.	Amount
RefundBalance	The total amount that has been refunded.	Amount

DebtBalance

Name	Description	Type
AvailableBalance	Available debt balance accumulated between recipient and sender.	Amount
PendingOutBalance	Any balance that is pending because of an external instrument was used to settle the debt.	Amount

DescriptorPolicy

For information about using the DescriptorPolicy type, see “Soft Descriptor Customization.”

Name	Description	Type
CSOwner	The recipient or caller customer service number. If you specify Caller, the customer service number for the caller is passed to the payment processor, which is the entity that actually processes payments on the person's credit card or bank account. Otherwise, the default value of CSOwner is Recipient.	The entity whose CS Phone number should be used. Valid values are either Recipient or Caller. For more information, see “Soft Descriptor Customization.” Default: Recipient
SoftDescriptorType	The type of soft descriptor. Valid values are	The type of soft

Name	Description	Type
	either Static or Dynamic. If you specify Static, or do not specify a type, the soft descriptor in your account level setting is sent to the payment processor. If you specify Dynamic, the first 15 characters of sender description is sent to the payment processor.	descriptor. Valid values are either Static or Dynamic. Default: Static

MarketplaceRefundPolicy

Name	Description	Type
MarketplaceTxnOnly	Caller refunds his fee to the recipient.	String
MasterAndMarketplaceTxn	Caller and Amazon FPS refund their fees to the sender, and the recipient refunds his amount	String
MasterTxnOnly	Caller does not refund his fee. Amazon FPS refunds its fee and the recipient refunds his amount plus the caller's fee to the sender.	String

RecipientVerificationStatus

Name	Description	Type
VerificationComplete	The account is verified to accept payments.	String
VerificationPending	The account is not verified. The customer needs to contact Amazon Payments to resolve the issue.	String
VerificationComplete NoLimits	The account is verified to receive funds from Amazon Payments and has no receiving limits.	String

RelatedTransaction

Name	Description	Type
RelationType	Relation type of the related transaction.	RelationType
TransactionId	The Transaction ID of the related transaction.	String Max size = 35 characters

StatusHistory

Name	Description	Type
Amount	The changed amount.	Amount
Date	The date when the status changed.	dateTime
StatusCode	The current status of the transaction.	String

Name	Description	Type
TransactionStatus	The current status of the transaction.	TransactionStatus

Token

Name	Description	Type
CallerReference	Account ID of the caller who initiated the original request.	String Max size = 128 Bytes
DateInstalled	The date and time when the payment token was created dateTime on the caller's account.	dateTime
FriendlyName	A name that references the token.	String Max size = 128 Characters
OldTokenId	The token ID linked to this token. The token that was created in place of this token.	String Size: 65 Bytes
PaymentReason	Payment reason passed during token installation.	String
TokenId	The token ID representing the payment instruction.	String Max size = 64 Characters
TokenStatus	Specifies whether or not the token is active.	TokenStatus
TokenType	The type of the token (e.g., single-use, multi-use, etc.).	TokenType

TokenUsageLimit

Name	Description	Type
Amount	Amount paid in the latest time window with this token.	Amount
Count	Number of times this token was used in the latest time Integer window.	Integer
LastResetAmount	Amount paid in the previous time window with this token.	Amount
LastResetCount	Number of times this token was used in the previous time window.	Integer
LastResetTimeStamp	The exact time when the latest time window started for this limit.	dateTime

Transaction

Name	Description	Type
CallerName	The value in this field is dependent on the account type. For a personal account, the contact name is displayed. For a business or developer account, the business name is displayed.	String Max size = 128 characters
CallerTransactionDate	Date the caller provided for the transaction.	dateTime
DateCompleted	Date the transaction was completed.	dateTime
DateReceived	Date the transaction was received by Amazon FPS.	dateTime
FPSFees	Amount of fees collected by Amazon FPS for performing the transaction.	Amount
FPSOperation	The operation type.	FPS Operation
OriginalTransactionId	In the case of a refund, the TransactionID that is being reversed.	String Max size = 35 characters
PaymentMethod	Payment method used in the transaction.	Payment Method
RecipientName	The value in this field is dependent on the account type. For a personal account, the contact name is displayed. For a business or developer account, the business name is displayed.	String Max size = 128 Characters
RecipientTokenID	The recipient token used in the transaction. Recipient tokens are needed when the caller and recipient are different people.	String Size: 65 Bytes
SenderName	The value in this field is dependent on the account type. For a personal account, the contact name is displayed. For a business or developer account, the business name is displayed.	String Max size = 128 characters
SenderTokenID	The sender token used in the transaction.	String Size: 65 Bytes
StatusCode	A code that represents the current status of the String transaction. Expands on the information in the TransactionStatus field. For example, if TransactionStatus is PENDING, this field might be PendingVerification, or PendingNetworkResponse.	String
StatusMessage	A short description of the current status of the transaction.	String
TransactionAmount	Total amount of the transaction.	Amount
TransactionId	Unique Amazon FPS-generated ID for the transaction.	String Max size = 35

Name	Description	Type
		Characters
TransactionPart	List of individual parts of the transaction, with each one dealing with your account's role in the transaction.	Transaction Part
TransactionStatus	Provides a short code on the status of the transaction, for example "PENDING."	Transaction Status

TransactionDetail

Name	Description	Type
CallerNamePDF	The value in this field is dependent on the account type. For a personal account, the contact name is displayed. For a business or developer account, the business name is displayed.	String Max size = 128 Characters
CallerDescription	Caller description the caller provided for the transaction.	String Constraint: Max size = 160 characters
CallerReference	Caller reference the caller provided for the transaction.	String Max size = 128 Characters
DateReceived	Date Amazon FPS received the transaction	dateTime
DateCompleted	Date the transaction was completed.	dateTime
FPSFees	Amount of fees collected by Amazon FPS for performing the transaction.	Amount
FPSFeesPaidBy	The party paying the FPS fees for this transaction	TransactionalRole
FPSOperation	The operation type.	FPSOperation
MarketPlaceFees	In the case of a marketplace transaction, this is the amount of any marketplace fee the caller has charged.	Amount
PaymentMethod	The payment method used.	PaymentMethod
RecipientEmail	The email ID of the recipient of this transaction.	String
RecipientName	The value in this field is dependent on the account type. For a personal account, the contact name is displayed. For a business or developer account, the business name is displayed.	String Max size = 128 characters
RecipientTokenId	Recipient token ID used in the transaction. Recipient tokens are needed when the caller and recipient are different people.	String Size: 65 Bytes
RelatedTransaction	All transactions related to this transaction.	RelatedTransaction
SenderDescription	Sender description the caller provided for	String

Name	Description	Type
	the transaction.	Constraint: Max size = 160 characters
SenderEmail	The email ID of the sender of this transaction. This String is returned only if the caller is also the recipient of this transaction.	String
SenderName	The value in this field is dependent on the account type. For a personal account, the contact name is displayed. For a business or developer account, the business name is displayed.	String Max size = 128 characters
SenderTokenId	Sender token ID used in the transaction.	String Size: 65 Bytes
StatusCode	A code that represents the current status of the String transaction.	String
StatusHistory	A list of all the previous status entries for this transaction.	StatusHistory
StatusMessage	A short description of the current status of the String transaction.	String
TransactionAmount	Total amount of the transaction.	Amount
TransactionId	Unique Amazon FPS-generated ID for the transaction.	String Max size = 35 characters
TransactionStatus	The transaction status.	TransactionStatus

TransactionPart

Name	Description	Type
Description	Description provided by the entity.	String
FeesPaid	Fees the caller or recipient paid.	Amount
InstrumentId	Payment instrument involved in this transaction part.	String
Name	Name used for the role specified in `Role`.	String
Reference	Reference data provided by this party.	String
Role	Role played by this party.	TransactionalRole

Co-Branded Service API Reference

You use different Co-Branded service APIs to create different payment tokens. For example, the Recurring-Use Token API creates a recurring-use payment token, whereas the Multi-Use Token API creates a multi-use payment token. The Advanced Quick Start also includes the Recipient Token API, which registers merchants with you (rather than creating a payment token) so they can receive payments. For more information about the different tokens, see “Multi-Use Payment Tokens”, “Recurring Payment Tokens”, and “Recipient Tokens.”

Common Parameters

The following parameters are common to all Co-Branded service API requests.

Request Parameters

Name	Description	Required
callerKey	AWS Access Key ID of the developer. You can obtain this value from the AWS Access Identifiers page on the AWS website (http://aws.amazon.com). Type: String Default: None	Yes
cobrandingStyle	Specifies the co-branding type on Amazon FPS payment authorization pages. Amazon FPS is phasing out support for the banner type, so we suggest you change your co-branding to the logo type. For more information, see “Co-Branding Styles.” Type: String Default: logo Valid Values: banner | logo	No
cobrandingUrl	Allows you to specify a co-branding URL dynamically. It specifies the URL of your company's logo. Type: String Default: None Constraint: This URL should point to a co-branding image that is not larger than 215 (w) x 40 (h) pixels in a secure HTTP server.	No
pipelineName	The kind of token you are creating. Type: String Default: None Valid Values: SingleUse | MultiUse | Recurring | Recipient | EditToken	Yes
returnURL	Specifies the URL on your website that the	Yes

Name	Description	Required
	person (typically the buyer) is redirected to after completing the CBUI web pages. In addition to the URL, the redirect URI includes the following: • Parameters appended to the returnURL in the URI • Status of the request • The installed token • The Signature Type: URL Default: None	
Signature	A value calculated using the request parameters and a SHA-1 HMAC encryption algorithm to make sure the request parameters and values were not altered during the request's or response's travel across the Internet. For more information, see “Working with Signatures.” Type: String Default: None	Yes
signatureVersion	2 **Important** The previous method for signing (signature version 1) was deprecated on November 3rd, 2009, and as of 10 February, 2011 it is no longer supported. Whenever you sign a request with your access keys, you must now use signature version 2.	Yes
signatureMethod	HmacSHA256 (preferred) or HmacSHA1	Yes
Version	The version of the API to use. Always set to 2009-01-09. Type: String	Yes
websiteDescription	Human readable text to describe your website. It is used on the payment authorization pages for messaging only. For instance, a message such as "Click here to return to <websiteDescription> website" can appear on the page. Type: String Default: None	No

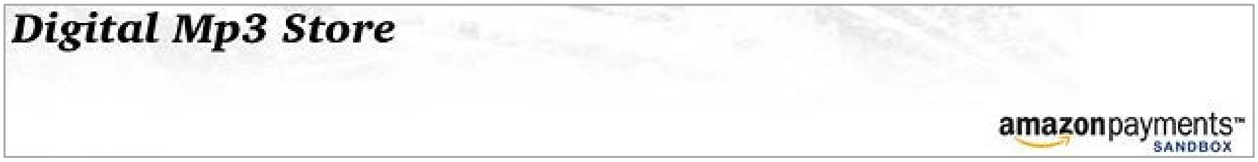
Co-Branding Styles

Co-branding refers to using your brand along with Amazon's on the CBUI pages. The CBUI offers the following co-branding styles.

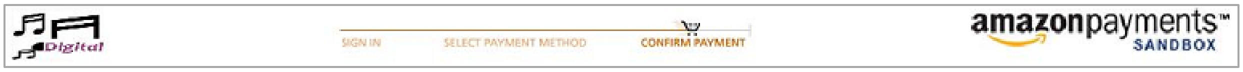
Banner—Your logo appears in the upper left corner of the CBUI page and the Amazon Payments logo appears right below your logo on the right hand side. Amazon is phasing out support for the banner type in favor of the logo type.

Logo—Your logo appears on the upper left corner of the CBUI page, followed by a checkout cart breadcrumb in the middle, followed by the Amazon Payments logo, as shown. This is the default behavior.

The following figure shows an example of banner co-branding.



The following figure shows an example of logo co-branding.



Response Parameters

The following table lists the parameters common to all Co-Branded service API responses.

Name	Description
certificateUrl	A url specifying the location of the certificate used for signing the response. Type: String Max Size: 1024 Bytes
Signature	A url specifying the location of the certificate used for signing the response. Type: String Max Size: 1024 Bytes
SignatureVersion	A value that specifies the Signature format. Type: Integer Valid Values: 2
SignatureMethod	A value that specifies the signing method. Type: String Valid Values: HmacSHA256 (preferred) and HmacSHA1.

Recipient Token API

Use this Co-Branded service API to register a recipient on a caller's website.

Request Parameters

Parameter	Description	Requirement
callerReference	A value you provide that uniquely identifies the request. For more information, see "Important Values to Store in Your Database." Type: String Constraint: Max size = 128 bytes	Yes
callerReferenceRefund	The caller reference used to identify the refund token. The recipient token pipeline installs a refund token that you can use to issue a refund on behalf of the recipient. Use callerReferenceRefund to retrieve the refund token using GetTokenByCaller. Type: String Constraint: Max size = 128 bytes	No
maxFixedFee	The maximum fixed fee that the caller charges in a marketplace transaction. The actual fixed marketplace fee is passed as a parameter to Pay or Reserve actions. This field can be ignored if a fixed fee is not being charged. Type: Long	No
maxVariableFee	The maximum variable fee that the caller charges in a marketplace transaction. The variable fee is a percentage of the transaction amount. The actual variable marketplace fee is passed as a parameter to Pay or Reserve actions. This field can be ignored if a variable fee is not being charged. Type: Long	No
paymentMethod	Specifies payment methods the recipient supports. Use CC for credit cards, ACH for bank account withdrawal, and ABT for Amazon Payments balance transfer. Type: Comma-separated list Default: ABT Valid Values: CC | ACH | ABT	No
recipientPaysFee	Set this value to True if the recipient agrees to pay the fees, otherwise set this value to False. Type: String	Yes

Parameter	Description	Requirement
	Valid Values: True | False	
validityExpiry	Specifies when the token expires. Use UNIX epoch date format. Type: Date Default: No expiry Constraint: Date cannot be earlier than the current date	No
validityStart	Specifies when the token becomes valid. Use UNIX epoch date format. Type: Date Default: Current date Constraint: Date must be within one year from the current date (date of creation) and cannot be earlier than the current date	No

Note

Co-Branded service request parameters are not case sensitive.

The request also uses the parameters common to all Co-Branded service API requests. For more information, see “Common Parameters.”

Parameter	Description
errorMessage	This is text in a human readable form that specifies the reason for a request failure. Type: String
Status	Specifies the status of the Co-Branded service request. Type: String Valid values: See the following table
tokenID	This string identifies the merchant who gets paid in the transaction. You should store this value. Type: String Size: 65 Bytes

Responses also include parameters common to all responses. For more information, see “Response Parameters.”

Status Code

The following table shows the values of the status response parameter.

Status Code	Description
SR	Success. Specifies that the merchant's token was created.
A	Specifies that the pipeline has been aborted by the user.
CE	Specifies a caller exception.
NP	There are four cases where the NP status is returned: • The payment instruction installation was not allowed on the sender's account, because the sender's email account is not verified • The sender and the recipient are the same • The recipient account is a personal account, and therefore cannot accept credit card payments • A user error occurred because the pipeline was cancelled and then restarted
NM	You are not registered as a third-party caller to make this transaction. Contact Amazon Payments for more information.

Recurring-Use Token API

The recurring-use token API creates a token that makes payments on a recurring basis at specific intervals. The sender's account is charged accordingly without requiring recurring authorizations. When the recurringPeriod is one month, the payment occurs on the same day each month. By default, if the original payment occurs on the last day of a month, the next payment will occur on the closest day to that without skipping a month. For example, if the pay date is October 31, the next pay date will be November 30. After the recurrence passes through February, the pay date is typically on the 28th of each month.

For example, if the first pay date is January 31st, your next two billing would typically be on February 28th and March 31st. Note, however, if your business requires it, the recurring-use token also allows you to set the next two payments to be February 28th and March 28th.

In another example, if the first payment occurs on March 5, the second payment can occur on or after April 5.

Request Parameters

Parameter	Description	Required
addressName addressLine1 addressLine2 city state	The sender's shipping address. You might choose to collect the address on your website and pass it to the CBUI. If you choose to collect	No

Parameter	Description	Required
country zip phoneNumber	the shipping address yourself, you can use these parameters to specify it. This address will be displayed to the sender on the payment authorization confirmation page. See also the description of collectShippingAddress. Type: String Default: None	
callerReference	A value you provide that uniquely identifies the request. For more information, see “Important Values to Store in Your Database.” Type: String Default: None Constraint: Max size = 128 bytes	Yes
collectShippingAddress	If you set this value to True, all the shipping address parameters (addressName, addressLine1, etc.) are ignored, and the shipping/mailing address that the sender confirms on the CBUI pages is returned as part of the return URL. Type: Boolean Default: False Valid Values: True | False	No
currencyCode	Specifies the currency of all amounts that this pipeline accepts. Type: String Default: USD Valid Values: USD	No
isRecipientCobranding	This parameter is for marketplace applications where the caller is different from the recipient. If this value is set to True, then the co-branding URL is picked up from the recipient's settings. The parameter recipientToken is mandatory if this value is set to True. If this value is False, the co-branding URL is picked up from the caller's	No

Parameter	Description	Required
	setting. Type: Boolean Valid Values: True | False Default: False	
paymentMethod	Specifies payment methods the recipient supports. Use CC for credit cards, ACH for bank account withdrawal, and ABT for Amazon Payments balance transfer. Type: Comma-separated list Default: ABT Valid Values: CC | ACH | ABT	No
paymentReason	Specifies the reason for this payment transaction. You can provide a limited set of HTML tags to format your text, including <b>, <i>, <u>, <ul>, <li>, , <em>, <strong>, and <strike>. Other tags are ignored. Type: String	No
recipientToken	Specifies the intended recipient's TokenId. Type: String Default: The caller is considered to be the recipient	No
recurringPeriod	The recurring period to associate with this token. This parameter accepts an integer followed by one of the following strings: • Hour[s] (e.g. 12 Hours) • Day[s] (e.g., 4 Days) • Month[s] (e.g., 6 Months) Type: String	Yes
transactionAmount	Specifies the recurring amount payable in this transaction. Type: String	Yes
validityExpiry	Specifies when the token expires. Use UNIX epoch date format. Type: Date Default: No expiration	No

Parameter	Description	Required
	Constraint: Date cannot be earlier than the current date	
validityStart	Specifies when the token becomes valid. Use UNIX epoch date format. Type: Date Default: The current date Constraint: The validity start date should be within one year from the current date. The date cannot be beyond one year or earlier than the current date.	No

The request also uses the parameters common to all Co-Branded service API requests. For more information, see “Common Parameters.”

Response Parameters

Parameter	Description
addressName addressLine1 addressLine2 city state zip phoneNumber	The sender's shipping address. These parameters are returned only if collectShippingAddress was set to True in the request. Type: String
errorMessage	This is text in a human readable form that specifies the reason for a request failure. Type: String
Expiry	The expiry (if any) of the payment method. Type: String Size: 20 Bytes
Status	The status of the Co-Branded service request. Type: String Valid Values: See “Status Codes”
tokenID	Specifies the token ID string associated with the token just created (installed). Type: String Size: 65 Bytes
warningCode	There might be cases when the sender token is installed successfully but there is an associated warning. This parameter denotes that warning. Type: String Valid Values: invalidShippingAddress (returned when you pass an incorrect shipping address in the request parameters addressLine1, addressLine2, city, etc.)

Parameter	Description
warningMessage	Specifies a human readable text that explains the warning corresponding to the warningCode. Type: String

Responses also include parameters common to all responses. For more information, see “Response Parameters.”

Status Codes

Status Code	Description
SA	Success status for the ABT payment method.
SB	Success status for the ACH (bank account) payment method.
SC	Success status for the credit card payment method.
SE	System error.
A	Buyer abandoned the pipeline.
CE	Specifies a caller exception.
PE	Payment Method Mismatch Error: Specifies that the buyer does not have payment method that you have requested.
NP	There are four cases where the NP status is returned: • The payment instruction installation was not allowed on the sender's account, because the sender's email account is not verified • The sender and the recipient are the same • The recipient account is a personal account, and therefore cannot accept credit card payments • A user error occurred because the pipeline was cancelled and then restarted
NM	You are not registered as a third-party caller to make this transaction. Contact Amazon Payments for more information.

Multi-Use Token API

The multi-use token API creates a token that makes multiple payments without the sender having to repeatedly authorize payments. The payments can occur at any time; they don't have to be on a regularly recurring basis. This token works like a voucher and the parameters you include in the request govern the token's use, such as how much the token can pay per transaction, how much it can pay during its lifetime, when it expires, what kind of personal payment instruments it can accept, and the minimum payment it can make.

You can specify a maximum amount that can be charged over a period of time. Set usageLimitType1 to Amount, and then use usageLimitPeriod1 and usageLimitValue1 to set the time period and maximum amount.

You can also specify a maximum number of charges that can occur over a period of time. Set usageLimitType2 to Count, and then use usageLimitPeriod2 and usageLimitValue2 to set the time period and maximum number of charges.

Request Parameters

Parameter	Description	Required
addressName addressLine1 addressLine2 city state country zip phoneNumber	The sender's shipping address. You might choose to collect the address on your website and pass it to the CBUI. If you choose to collect the shipping address yourself, you can use these parameters to specify it. This address will be displayed to the sender on the payment authorization confirmation page. See also the description of collectShippingAddress. Type: String Default: None	No
amountType	Specifies whether the amount specified by transactionAmount is the exact amount allowed by the token, the minimum amount, or the maximum amount. Type: String Default: Exact Valid Values: Exact | Maximum | Minimum	No
callerReference	A value you provide that uniquely identifies the request. For more information, see “Important Values to Store in Your Database.” Type: String Default: None Constraint: Max size = 128 bytes	Yes
collectShippingAddress	If you set this value to True, all the shipping address parameters (addressName, addressLine1, etc.) are ignored, and the shipping/mailing address that the sender confirms on the CBUI pages is returned as part of the return URL. Type: Boolean Default: False Valid Values: True | False	No
currencyCode	Specifies the currency of all amounts that this pipeline accepts. Type: String Default: USD Valid Values: USD	No
globalAmountLimit	Specifies the maximum amount of money this token can be used for across all payments. Type: String	Yes
isRecipientCobranding	This parameter is for marketplace applications where the caller is different from the recipient.	No

Parameter	Description	Required
	If this value is set to True, then the co-branding URL is picked up from the recipient's settings. The parameter recipientToken is mandatory if this value is set to True. If this value is False, the co-branding URL is picked up from the caller's setting. Type: Boolean Valid Values: True | False Default: False	
paymentMethod	Specifies payment methods the recipient supports. Use CC for credit cards, ACH for bank account withdrawal, and ABT for Amazon Payments balance transfer. Type: Comma-separated list Default: ABT Valid Values: CC | ACH | ABT	No
paymentReason	Specifies the reason for this payment transaction. You can provide a limited set of HTML tags to format your text, including <b>, <i>, <u>, <ul>, <li>, , <em>, <strong>, and <strike>. Other tags are ignored. Type: String	No
recipientTokenList	Specifies a comma-separated list of recipient token IDs (for the merchants who can receive payment with this token). If you create a merchant token list, you must include the recipient token ID for the caller in addition to any other tokens you add to the list. Type: String Default: Caller's recipient token ID Conditional: Required if isRecipientCobranding is True	Conditional
transactionAmount	Specifies the amount payable in this transaction. Type: Double Condition: Required if you have specified amountType.	Conditional
usageLimitType1	See the description at the beginning of this topic for how to use this field. Type: String Valid Values: Amount | Count	No
usageLimitPeriod1	If you specify usageLimitType1 but not usageLimitPeriod1, the usage period is forever. This parameter accepts an integer followed by one of the following strings: • Hour[s] (e.g., 12 Hours) • Day[s] (e.g., 4 Days)	No

Parameter	Description	Required
	- Month[s] (e.g., 6 Months) Type: String Default: Forever	
usageLimitValue1	Corresponds to the value for the parameter specified by usageLimitType1. Type: String if usageLimitType1=Amount; integer if usageLimitType1=Count Condition: Required if you specify usageLimitType1	Conditional
usageLimitType2	See the description at the beginning of this topic for how to use this field. Type: String Valid Values: Amount | Count	No
usageLimitPeriod2	If you specify usageLimitType2 but not usageLimitPeriod2, the usage period is forever. This parameter accepts an integer followed by one of the following strings: - Hour[s] (e.g., 12 Hours) - Day[s] (e.g., 4 Days) - Month[s] (e.g., 6 Months) Type: String Default: Forever	No
usageLimitValue2	Corresponds to the value for the parameter specified by usageLimitType2. Type: String if usageLimitType2=Amount; integer if usageLimitType2=Count Condition: Required if you specify usageLimitType2	Conditional
validityExpiry	Specifies when the token expires. Use UNIX epoch date format. Type: Date Default: If a value is not provided for this parameter, the expiration defaults to one year. Constraint: Date cannot be earlier than the current date	No
validityStart	Specifies when the token becomes valid. Use UNIX epoch date format. Type: Date Default: The current date Constraint: The validity start date should be within one year from the current date. The date cannot be beyond one year or earlier than the current date.	No

The request also uses the parameters common to all Co-Branded service API requests. For more information, see “Common Parameters.”

Response Parameters

Parameter	Description
addressName addressLine1 addressLine2 city state zip phoneNumber	The sender's shipping address. These parameters are returned only if collectShippingAddress was set to True in the request. Type: String
errorMessage	This is text in a human readable form that specifies the reason for a request failure. Type: String (LOB, Large Object)
Expiry	Specifies the expiry (if any) of the payment method. Type: String Size: 20 Bytes
Status	The status of the Co-Branded service request. Type: String Valid Values: See “Status Codes.” Size: 2 Bytes
tokenID	Specifies the token ID string associated with the token just created (installed). Type: String Size: 65 Bytes
warningCode	There might be cases when the sender token is installed successfully but there is an associated warning. This parameter denotes that warning. Type: String Valid Values: invalidShippingAddress (returned when you pass an incorrect shipping address in the request parameters addressLine1, addressLine2, city, etc.) Size: 64 Bytes
warningMessage	Specifies a human readable text that explains the warning corresponding to the warningCode. Type: String (LOB, Large Object)

Responses also include parameters common to all responses. For more information, see “Response Parameters.”

Status Codes

Status Code	Description
SA	Success status for the ABT payment method.
SB	Success status for the ACH (bank account) payment method.
SC	Success status for the credit card payment method.
SE	System error.
A	Buyer abandoned the pipeline.
CE	Specifies a caller exception.
PE	Payment Method Mismatch Error: Specifies that the buyer does not have the payment method you requested.
NP	There are several cases where the NP status is returned: • The payment instruction installation was not allowed on the sender's account, because the sender's email account is not verified • The sender and the recipient are the same • The recipient account is a personal account, and therefore cannot accept credit card payments • A user error occurred because the pipeline was cancelled and then restarted • The account associated with one of the recipient tokens you specified is closed
NM	You are not registered as a third-party caller to make this transaction. Contact Amazon Payments for more information.

Edit Token API

The Edit Token API enables you to view an existing token's details and to change the payment instrument for the token.

Request Parameters

Parameter	Description	Required
callerReference	A value you provide that uniquely identifies the request. For more information, see “Important Values to Store in Your Database.” Type: String Default: None Constraint: Max size = 128 bytes. Must be unique among your payment tokens.	Yes
paymentMethod	A comma-separated list that enables you to pass payment methods supported by the merchant. Type: String Default: ABT	No

Parameter	Description	Required
	Valid Values: CC | ACH | ABT	
tokenID	Specifies the token you want to edit. You can only edit multi-use or recurring tokens. Type: String Constraint: Max size = 65 characters	Yes

The request also uses the parameters common to all Co-Branded service API requests. For more information, see “Common Parameters.”

Response Parameters

Parameter	Description
errorMessage	This is text in a human readable form that specifies the reason for a request failure. Type: String
Expiry	This specifies the expiry (if any) of the payment method associated with the token. Type: String Size: 20 Bytes
Status	The status of the Co-Branded service request. Type: String Valid Values: See “Status Codes.”
tokenID	This specifies the token ID associated with the new token. Type: String Size: 65 Bytes

Responses also include parameters common to all responses. For more information, see “Response Parameters.”

Status Codes

Status Code	Description
SU	The token has not changed.
SA	The status of the token that uses ABT as the payment method has changed.
SB	The token that uses ACH as the payment method has changed.
SC	The token that uses CC as the payment method has changed.
SE	System error.
A	Buyer abandoned the pipeline.
CE	Specifies a caller exception.
IT	Specifies an invalid token. The token passed is invalid, expired, or cannot be modified.
NA	Specifies that the user account does not exist.
UT	Specifies an unauthorized access to the token. The token does not belong to

Status Code	Description
	the user.
PE	Specifies a payment method mismatch error. The user does not have payment method that you requested.

Code Samples

The following sections provide information about the Amazon Flexible Payments Service (FPS) development libraries and sample code provided by Amazon. The sample code shows you how to implement most of the basic Amazon FPS functions. Packaged in four programming languages (C#, Java, Perl, and PHP), the development libraries are available from the Amazon Web Services developer community, under the Amazon Flexible Payments Service category. Refer to the following table for specific sample packages.

Language	Location
C#	Amazon FPS Quick Starts (API version - 2010-08-28): Standalone Library in C#
Java	Amazon FPS Quick Starts (API version - 2010-08-28): Standalone Library in Java
Perl	Amazon FPS Quick Starts (API version - 2010-08-28): Standalone Library in Perl
PHP	Amazon FPS Quick Starts (API version - 2010-08-28): Standalone Library in PHP

Each package is updated for signature version 2, and contains both a development library and a collection of sample implementations of the Amazon FPS APIs. The development libraries enable you to

- Use the Co-Branded User Interface to create CBUI pipeline URLs
- Invoke any of the Amazon FPS APIs documented in this quick start
- Generate signatures compliant with signature version 2
- Validate the content of return URL responses and IPN notifications

For help building your first sample application using the development libraries, see “Making a Pay Request” in the Amazon Flexible Payments Service Getting Started Guide.

Understanding the Amazon FPS Samples

Amazon provides dozens of samples in four programming languages (C#, Java, Perl, and PHP) which show you how to perform numerous operation with Amazon FPS actions.

When you download a sample file, such as amazon-fps-2008-09-17-java-library, the [package root]/src/com/amazonaws/fps/samples folder contains sample classes showing how to invoke most Amazon FPS actions from your code ([package root] is the location you extracted your sample package).

Each sample describes its requirements in its Readme.html file, located at the package root. Typically, the entire library structure must be available to the compiler. For example, the amazon-fps-2008-09-17-php-library.zip file contains the src/Amazon/FPS/Model and src/Amazon/FPS/Mock folders, which the files in src/Amazon/FPS/Samples require.

In addition to these primary components, a sample may include other required resources. For example, the Java samples all include numerous jar files in the [package-root]/third-party folder, which must also be in your classpath in order to compile the sample.

For each sample, you must set your security credentials and Amazon FPS sandbox endpoints in a library-dependant way. For example, to use the C# library, you set your security credentials in the [package-root]/src/Amazon.FPS.Samples/Amazon.FPS.Samples/AmazonFPSSamples.cs file, while for the perl library you set them in the individual [package-root]/src/Amazon/FPS/Samples/*.pl file you are working with.

In the following section, we show how to work with the VerifySignature sample using the Java library. You will use this fundamental API frequently for server-side validation of your return URL responses and IPN notifications.You will find that the basic process you use for the VerifySignature sample is the same for all the other samples in the FPS/Samples (or, in the case of Amazon.FPS.Samples) folder. (The process for the CBUI and Return URL/IPN Validations samples are different. For more information, see “Understanding the Amazon CBUI Samples.”)

Understanding the VerifySignature Sample

This section explains how to use the Java version of the VerifySignature API. If you want to use one of the other sample libraries, they are set up nearly identically to the Java sample. To see file locations for the VerifySignature sample for your preferred language, see “Locations of the VerifySignatureSample Files in Other Libraries.”

To use the sample, do the following

Using the VerifySignature Sample

1. Set up your programming environment so that the program will compile without warnings or errors. For the Java sample, this includes ensuring that the files and sub folders in the [package-root]/src and [package-root]/third-party folder are in the java classpath.
2. In the [package-root]/src/config.properties file, set the values for AwsAccessKey and AwsSecretKey using your security credentials.

> **Important**
> Your Secret Access Key is a secret, which only you and Amazon should know. It is important to keep it confidential to protect your account. Store it securely. Never include it in your requests to the Amazon Flexible Payments Service (Amazon FPS), and never email it to anyone. Do not share it outside your organization, even if an inquiry appears to come from Amazon Web Services (AWS) or Amazon.com. No one who legitimately represents Amazon will ever ask you for your Secret Access Key.

To get your credentials, see “Getting an AWS Account” in the Amazon Simple Pay Getting Started Guide.
In the same file, if you want to target the sandbox, change the AwsServiceEndPoint property to https://fps.sandbox.amazonaws.com. Then save the file.

3. In the [package-root]/src/com/amazonaws/fps/samples/VerifySignature.java file, find the section containing the lines:

```
VerifySignatureRequest fpsRequest = new VerifySignatureRequest();
// @TODO: set request parameters here
// invokeVerifySignature(service, fpsRequest);
```

(The VerifySignatureRequest, VerifySignatureResult, and VerifySignatureResponse classes are located in [package-root]/src/com/amazonaws/fps/model folder.)

4. In the same file, remove the comment on invokeVerifySignature, and after it add the VerifySignature parameter assignments consistent with your transaction. For example:

```
fpsRequest.setAction("VerifySignature");
fpsRequest.setUrlEndpoint("http://myApplication/my-ipn-
response.pgp");
fpsRequest.setHttpParameters(
"Name1=Joe&
"Name2=College&" +
"signatureVersion=2&" +
"signatureMethod=HMACSHA256&" +
"certificateUrl=https://fps.amazonaws.com/cert/key.pem&" +
"signature=aoeuAOE123eAUdhf]");
```

Save the file. For information on the parameters to VerifySignature, see “VerifySignature.”

5. Compile and run the sample.
The program copies to standard out a representation of the VerifySigatureResponse XML fragment similar to the following:

```
VerifySignature Action Response
=======================================
VerifySignatureResponse
VerifySignatureResult
True
VerificationStatus
Success
ResponseMetadata
RequestId
bda6-4f5f-b37b-1a146b9a-b9e45c3012a5:0
```

For information on the XML document returned by VerifySignature, see “VerifySignature”).

In addition to simple API invocation, the samples provide you the following advanced options:

- The ability to simulate a mock Amazon FPS service and get responses without a live connection.
- Specifying a proxy host and port, through config.properties.
- Setting the endpoint, through config.properties.
- Logging, through log4j.properties.

Locations of the VerifySignatureSample Files in Other Libraries

The development libraries for C#, Perl, and PHP also enable you to perform a server-side validation of a signature in a return URL or IPN notification. The following tables list the locations of the files referenced in Understanding the Amazon FPS Samples.

C# File Locations for the Amazon.FPS VerifySignature Sample

File	Location
VerifySignatureRequest.cs	[package root]/src/ Amazon.FPS/Amazon.FPS.Model
VerifySignatureSample.cs	[package root]/src/ Amazon.FPS.Samples/Amazon.FPS.Samples
Amazon.FPS.proj (Visual Studio.NET project for the FPS development library)	[package root]/src/Amazon.FPS/Amazon.FPS
Amazon.FPS.Samples.proj (Visual Studio.NET project for the Amazon FPS API samples)	[package root]/src/Amazon.FPS/ Amazon.FPS.Samples/Amazon.FPS.Samples
Amazon.FPS.sln Visual Studio.NET solution for the library package **Note** The Visual Studio.NET samples are organized into this solution. After setting your access parameters the first time, you build the entire solution to generate the dependency classes. Then you modify the specific sample you want. See the Readme.html file for more information.	[package root]/src/Amazon.FPS/Amazon.FPS

Perl File Locations for the Amazon.FPS VerifySignature Sample

Class	Location
VerifySignatureRequest.pm	[package root]/src/Amazon/FPS/Model
VerifySignatureSample.pl	[package root]/src/Amazon/FPS/Samples
ReadMe.html (readme for perl fps library)	[package root]/src

PHP File Locations for the Amazon.FPS VerifySignature Sample

Class	Location
VerifySignatureRequest.php	[package root]/src/Amazon/FPS/Model
VerifySignatureSample.php	[package root]/src/Amazon/FPS/Samples
ReadMe.html (readme for php fps library)	[package root]/src

Understanding the Amazon CBUI Samples

Amazon provides five samples in four programming languages (C#, Java, Perl, and PHP) which show you how to build Co-Branded User Interface request URLs.

When you download a sample file, such as amazon-fps-2008-09-17-java-library, the [package root]/src/com/amazonaws/cbui/samples folder contains sample classes showing how to generate pipeline-specific CBUI URLs from your code ([package root] is the location you extracted your sample package).

Each sample describes its requirements in its Readme.html file, located at the package root. Typically, the entire library structure must be available to the compiler. For example, the amazon-fps-2008-09-17-php-library.zip file contains the src/Amazon/CBUI and folders which the files in src/Amazon/CBUI/Samples require.

In addition to these primary components, a sample may include other required resources. For example, the Java samples all include numerous jar files in the [package-root]/third-party folder, which must also be in your classpath in order to compile the sample.

For each sample, you must set your security credentials and Amazon FPS sandbox endpoints in a library-dependant way. For example, to use the C# library, you set your security credentials in the [package-root]/src/Amazon.FPS.Samples/Amazon.FPS.Samples/ AmazonFPSSamples.cs file, while for the perl library you set them in the individual [package-root]/src/Amazon/CBUI/Samples/*.pl file you are working with.

The following samples are provided with each sample library:

Class	Location
CBUI Single Use Pipeline Sample	Requests authorization for a one-time payment.
CBUI Multi-Use Pipeline Sample	Requests authorization for multiple payments.
CBUI Recipient Pipeline Sample	Requests authorization for a recipient token pipeline, such as that needed for marketplace fixed and variable fees.
CBUI Recurring Token Pipeline Sample	Requests authorization for a recurring token pipeline, such as that needed for periodic charges.
CBUI Edit Token Pipeline Sample	Requests authorization for an edit-token pipeline.

In the following section, we show how to work with the CBUI Single Use Pipeline sample using the Java library. This sample enables you to set up a single-use token for a one-time payment. You will find that the basic process you use for the CBUI Single Use Pipeline sample is the same for all the other samples in the CBUI/Samples (or, in the case of Amazon.CBUI.Samples) folder. (The process for the FPS and Return URL/IPN Validations samples are different. For more information, see “Understanding the Amazon FPS Samples.”

Java

This section describes the Java version of the CBUISingleUsePipeline. The files for the C#, Perl, and PHP CBUISingleUsePipeline samples are listed in Locations of the CBUISingleUsePipeline Files in Other Libraries.

The CBUISingleUsePipeline sample centers on the following files:

File	Description
config.properties	Set your AWS access key ID, AWS secret key, and sandbox endpoint in this file. Important Your Secret Access Key is a secret, which only you and Amazon should know. It is important to keep it confidential to protect your account. Store it securely. Never include it in your requests to the Amazon Flexible Payments Service (Amazon FPS), and never email it to anyone. Do not share it outside your organization, even if an inquiry appears to come from AmazonWeb Services (AWS) or Amazon.com. No one who legitimately represents Amazon will ever ask you for your Secret Access Key.
CBUISingleUsePipelineSample.java	In the main method, you create an AmazonFPSSingleUsePipeline object and use it to add parameters specific to your application.
AmazonFPSSingleUsePipeline.java	Invoked from CBUISingleUsePipelineSample.java, this class contains the setMandatoryParameters and validateParameters functions which you can customize for your application.

Co-Branded service request with Java SDK Sample

1. Open the file [package-root]/src/config.properties, and set AwsAccessKey and AwsSecretKey properties to your AWS access key and AWS secret key, respectively.

   Important
   Your Secret Access Key is a secret, which only you and Amazon should know. It is important to keep it confidential to protect your account. Store it securely. Never include it in your requests to the Amazon Flexible Payments Service (Amazon FPS), and never email it to anyone. Do not share it outside your organization, even if an inquiry appears to come from Amazon Web Services (AWS) or Amazon.com.No one who legitimately represents Amazon will ever ask you for your Secret Access Key.

   To get your security credentials, see “Getting an AWS Account” in the Amazon Flexible Payments Service Getting Started Guide.
2. In the same file, set the AwsServiceEndPoint to https://fps.sandbox.amazonaws.com/ (the Amazon FPS sandbox).
3. In the same file, set the CBUIServiceEndPoint to https://authorize.payments-sandbox.amazon.com/cobranded-ui/actions/start (the Co-Branded service sandbox).
4. Open the file [package-root]/src/com/amazonaws/cbui/samples/CBUISingleUsePipelineSample.java, and find the following line:

   ```
   AmazonFPSSingleUsePipeline pipeline= new
   AmazonFPSSingleUsePipeline(accessKey, secretKey);
   ```

   Change the pipeline.setMandatoryParameters and pipeline.addParameters method calls to the following:

   ```
   //pipeline name, your return URL, and the amount
   pipeline.setMandatoryParameters("callerReferenceSingleUse",
   "[your returnUrl]", "5");
   //optional parameters pipeline.addParameter("currencyCode",
   "USD");
   pipeline.addParameter("paymentReason", "Now and Forever – Richard
   Mark");
   pipeline.addParameter("paymentMethod", "ABT,ACH,CC">;
   pipeline.addParameter("callerReference", "[Unique ID for the
   transaction]");
   ```

   Save the file.
5. Ensure that all the jar files in the third-party folder and sub folders are in your java CLASSPATH.
6. Compile and run the sample. The Co-branded authorization page is printed to standard out.
7. Using a web browser, navigate to the URL produced by the sample. Because the sender and recipient cannot be the same, you must use an account different from your AWS developer or business accounts.

8. When complete, the page you specified as [your returnUrl] is hit with the Co-Branded service response. You validate this response by testing the signature.

You can customize the sample by modifying the file [package-root]/src/com/amazonaws/cbui/AmazonFPSSingleUsePipeline.java. The setMandatoryParameters only requires callerReference, returnUrl, and transactionAmount. If you want to make more parameters mandatory, modify this method.

In the same file, the validateParameters function ensures that the transactionAmount parameter is present. You can add custom validation checks to this method.

Locations of the CBUISingleUsePipeline Files in Other Libraries

The development libraries for C#, Perl, and PHP also enable you to create CBUI pipeline urls. The following tables indicate the locations of the files referenced in Understanding the Amazon CBUI Samples.

C# File Locations for the Amazon.FPS CBUI Sample

File	Location
CBUISingleUsePipelineSample.cs	[package root]\src\Amazon.CBUI\ Amazon.CBUI.Model.
AmazonFPSSingleUsePipeline.cs	[package root]\src\Amazon.CBUI.Samples\ Amazon.CBUI.Samples
Amazon.CBUI.proj (Visual Studio.NET solution for the development library)	[package root]\src\Amazon.CBUI\Amazon.CBUI\
Amazon.CBUI.Samples.proj (Visual Studio.NET solution for the Amazon FPS API samples)	[package root]\src\Amazon.CBUI\ Amazon.CBUI.Samples\ Amazon.CBUI.Samples
Amazon.FPS.sln Visual Studio.NET solution for the library package **Note** The Visual Studio.NET samples are organized into this solution. After setting your access parameters the first time, you build the entire solution to generate the dependency classes. Then you modify the specific sample you want. See the Readme.html file for more information.	[package root]/src/Amazon.FPS/ Amazon.FPS

Perl File Locations for the Amazon.FPS VerifySignature Sample

Class	Location
CBUISingleUsePipelineSample.pl	[package root]/src/Amazon/CBUI/Samples.
AmazonFPSSingleUsePipeline.pm	[package root]/src/Amazon/CBUI
ReadMe.html (readme for perl FPS library)	[package root]/src

PHP File Locations for the Amazon.FPS VerifySignature Sample

Class	Location
CBUISingleUsePipelineSample.php	[package root]/src/Amazon/CBUI/Model.
CBUISingleUsePipeline.php	[package root]/src/Amazon/CBUI/Samples
ReadMe.html (readme for php fps library)	[package root]/src

Understanding the IPNAndReturnURLValidation Sample

Amazon provides samples in four programming languages which show you how to perform a server-side verification of the signatures in both the return URL and in IPN notifications. In this section, we will briefly go over the essential details of the Java version only. The other samples differ only in the programming language used for rendering them. For specific comprehensive information on a particular sample, see its IPNAndReturnURLValidation.html file.

Each IPNAndReturnURLValidation sample contains three primary components in the src/com/amazonaws/ipnreturnurlvalidation folder. These are:

File	Description
ReturnUrlVerificationSampleCode.java	This class contains the program entry point for verifying the signature contained in a return URL, and thereby validating the return URL content. It sets up initial parameter values for return URL responses, and then calls the static method SignatureUtilsForOutbound .validateRequest with those values.
IPNVerificationSampleCode.java	This class contains the program entry point for verifying the signature contained in an IPN notification. It sets up initial parameter values for IPN notifications, and then calls the static method SignatureUtilsForOutbound .validateRequest with those values.
SignatureUtilsForOutbound.java	Invoked from ReturnUrlVerificationSampleCode.java and IPNVerificationSampleCode.java, this class uses the signature version 2 process to validate the signature. It contains methods to reassemble the string to sign, URL encode the

File	Description
	string, and sign it using the Amazon certificate listed as the signer. Finally, it validates the signature and prints the result to standard out.

In addition to these primary components, a sample may include other required resources. For example,the Java samples all include the `third-party` folder, the jar files of which must be in your classpath in order to compile the sample.

To use the sample, do the following

Using the IPNAndReturnURLValidation Sample

1. Set up your programming environment so that the program will compile without warnings or errors. For the Java sample, this includes ensuring that the src/com/amazonaws/ipnreturnurlvalidation folder and the files are available to the compiler, either by including them as command line parameters, or, if you build using an IDE, by including them as project resources.
2. The ReturnUrlVerificationSampleCode and IPNVerificationSampleCode classes use a HashMap to store parameters which correspond to the fields returned during a return URL response or an IPN notification. Modify these values to suit the response you want to validate. These are the only values you need to change using this sample.
3. Compile the sample. For example, if you are including the [package-root]src/third-party/commons-codec-1.3/commons-codec-1.3.jar using the linux command line, you would type

   ```
   $javac -cp .:[package-root]
   src/third-party/commons-codec-1.3/commons-codec-1.3.jar
   ReturnUrlVerificationSampleCode.java
   SignatureUtilsForOutbound.java
   ```

   On Windows, you would type

   ```
   $javac -cp .;[package-root]
   src/third-party/commons-codec-1.3/commons-codec-1.3.jar
   ReturnUrlVerificationSampleCode.javaSignatureUtilsForOutbou
   nd.java
   ```
4. Run the sample. Continuing the previous example, on linux, you would type

   ```
   $javac -cp .:[package-root]
   src/third-party/commons-codec-1.3/commons-codec-1.3.jar
   ReturnUrlVerificationSampleCode
   ```

   On Windows, you would type

   ```
   $javac -cp .;[package-root]
   src/third-party/commons-codec-1.3/commons-codec-1.3.jar
   ReturnUrlVerificationSampleCode
   ```

The result "**Is signature correct: true**" is printed to standard out if the verification determines the signature to be valid.

Locations of the IPNAndReturnURLValidation Files in Other SDKs

The development libraries for C#, Perl, and PHP also enable you to test Return URL and IPN notifications. The following tables indicate the locations of the files referenced in Understanding the IPNAndReturnURLValidation Sample.

C# File Locations for the Amazon.IpnReturnUrlValidationSample Library

File	Location
ReturnUrlVerificationSampleCode.cs	[package root] src\Amazon.IpnReturnUrlValidation\.
IPNVerificationSampleCode.cs	[package root] src\Amazon.IpnReturnUrlValidationSamples. IpnReturnUrlValidationSamples\
SignatureUtilsForOutbound.cs	[package root] src\Amazon.IpnReturnUrlValidationSamples. IpnReturnUrlValidationSamples\
IpnAndReturnUrlValidation.html (readme for this sample)	[package root] src\Amazon.IpnReturnUrlValidationSamples. IpnReturnUrlValidationSamples\
IpnReturnUrlValidation.Samples.csproj (Visual Studio.NET project for this sample)	[package root] src\Amazon.IpnReturnUrlValidation\

Perl File Locations for the IpnReturnUrlValidation Library

Class	Location
ReturnUrlVerificationSampleCode.pl	[package root] src/Amazon/IpnReturnUrlValidation/Samples.
IPNVerificationSampleCode.pl	[package root] src/Amazon/IpnReturnUrlValidation/Samples
SignatureUtilsForOutbound.pm	[package root] src/Amazon/IpnReturnUrlValidation
IpnAndReturnUrlValidation.html [package root] src (readme for this sample)	[package root] src

PHP File Locations for the IpnReturnUrlValidation Library

Class	Location
ReturnUrlVerificationSampleCode.php	[package root]src/ Amazon/IpnReturnUrlValidation/ Samples.
IPNVerificationSampleCode.php	[package root]src/ Amazon/IpnReturnUrlValidation/Samples
SignatureUtilsForOutbound.php	[package root]src/ Amazon/IpnReturnUrlValidation
IpnAndReturnUrlValidation.html [package root] src (README for this sample)	[package root] src

Getting the Samples

The Amazon FPS sample applications are available from the Amazon Web Services developer center.

To download Amazon FPS samples:

1. Go to http://developer.amazonwebservices.com/connect/forumindex.jspa.
   The **Discussion Forums** page opens.
2. From the **Developers** menu, choose **Sample Code & Libraries**.
3. In the **Browse by Category** area, choose **Amazon Flexible Payments Service**.
4. Choose your sample of interest in the programming language you prefer. To obtain the sample applications listed in this guide, look for sample applications whose package name resembles the format "amazon-fps-2008-09-17-*LANGUAGE*-library." For example, the Java sample is available in the file **amazon-fps-2008-09-17-java-library.zip**.
5. Read the instructions on the page. Note that this page enables you to start a community discussion about sample.You can also review it. When you are ready to proceed, click **Download**.
   The **Opening Amazon** window opens. Ensure it is the sample you want, and Click **OK**
6. Extract the zipped files to a convenient location on your workstation.

Each download includes sample-specific instructions in its README.txt file. For general guidance on the samples applicable to this edition of Amazon FPS, see “Code Samples.”

Amazon FPS Resources

The following table lists related resources that you'll find useful as you work with this service.

Resource	Description
Amazon Flexible Payments Service Getting Started Guide	Gets you set up with Amazon FPS, and shows you how to implement a simple one-time payment using Amazon FPS Basic Quick Start.
Amazon Flexible Payments Marketplace Quick Start	Covers the marketplace functionality of Amazon FPS.
Amazon Flexible Payments Service Advanced Quick Start	Covers the multiple-payment functionality of Amazon FPS.
Amazon Flexible Payments Service Account Management Quick Start	Covers the account management functionality of Amazon FPS.
FAQs	Frequently asked questions about using Amazon FPS.
Release Notes	Provides a high-level overview of the current release, noting any new features, corrections, and known issues.
FPS Developer Resource Center	A starting point specifically for FPS documentation, code samples, release notes, and other information to help you build innovative applications.
Discussion Forums	A community-based forum for developers to discuss technical questions related to Amazon FPS.
Product information about Amazon FPS	The primary web page for information about Amazon FPS.
AWS Developer Resource Center	A central starting point for AWS documentation, code samples, release notes, and other information to help you build innovative applications.
AWS Support Center	The home page for AWS Technical Support, including access to our Developer Forums, Technical FAQs, Service Status page, and Premium Support (if you are subscribed to this program).
Contact Us	A central contact point for inquiries concerning AWS billing, accounts, events, abuse, and more.
Conditions of Use	Detailed information about Amazon.com copyright and trademark usage and other topics.

Glossary

ABT, Amazon Payments Account Balance	One method of payment available with Amazon Payments.
access key rotation	For added security, you can switch between an active and inactive access key on your AWS security credentials page.
AWS Access Key ID	A string distributed by AWS that uniquely identifies your AWS developer account. You include this ID in every request.
ACH, Bank Account Debits	One method of payment available with Amazon Payments.
buyer	The buyer pays the seller for a product or service.
caller	A developer who facilitates payment between a sender and a recipient.
chargeback	a reversal of a payment issued by the bank when the buyer disputes the charge.
Co-Branded User Interface (CBUI)	A set of Amazon Payments web pages which lead a user through a secure login and payment authorization pipeline. Once the pipeline is complete, the user is redirected to your website.
endpoint	The URI that specifies the destination of an API request.
HMAC	The Hash Message Authentication Code used to authenticate a message. The HMAC is calculated using a standard, hash cryptographic algorithm, such as SHA-256. This algorithm uses a key value to perform the encryption. That key is your Secret Key. For that reason, your Secret Key must remain a shared secret between you and Amazon Payments.
inbound requests	Button click or other form request to Amazon Payments. Also inbound notification.
Instant Payment Notification	A notification that is sent whenever a payment, refund, or reserved payment completes successfully or fails. The caller must host this notification service and provide Amazon Payments with its URL.
marketplace, marketplace scenario	An environment in which the caller charges a fee for facilitating a transaction between a sender and a

recipient.

order pipeline

The steps through which an order passes between the time a customer selects an item and the customer's pay instrument is charged.

outbound notifications

Response from Amazon Payments to your Amazon FPS application by way of Return URL or IPN.

payment instrument

The method of payment a customer chooses to use with Amazon Payments. These are credit cards, Amazon Payments account balance (ABT), and bank account debits (ACH)

one time payment

An Amazon FPS payment processed with a single-use payment token. When the payment is made, the token may no longer be used.

recipient

A seller who receives a payment from a buyer (sender) in exchange for a service or product.

Recipient Token

Payment token created when a seller authorizes a payment of marketplace fees to you for hosting services, often with a Register Now button.

recurring payment

An Amazon FPS payment processed with a recurring payment token. Payments are made periodically using the same payment token. The token is valid until it expires.

reserve

The amount that is put in reserve against a credit card but not charged. Later, the transaction is settled (typically when the product is actually shipped).

sandbox

A part of the Amazon Payments web service where you can test the functionality of your application without incurring charges or purchasing products.

Secret Key

A string distributed by AWS that uniquely identifies your AWS developer account. The Secret Key is a shared secret between the developer and AWS. The Secret Key is used as the key in the HMAC algorithm that encrypts the signature.

seller

The seller receives money from a buyer in exchange for a service or product.

sender

The sender (also known as the buyer) pays a recipient for a product or service.

settle

To complete a transaction that has been reserved. If you don't charge the sender immediately upon the initiation of the purchase (and instead reserve the amount against the sender's credit card), you settle the transaction later, typically after you ship the product to the sender. Settle actually makes the reserved amount move from the sender to the recipient.

SHA1, SHA256

Secure Hash Algorithms used for Amazon Web Services signatures. SHA1 is an earlier version of the algorithm, which is currently being deprecated for Amazon Web Services. SHA256 is its more secure replacement.

signature

A URL-encoded string composed of request parameters and their values encrypted using an HMAC algorithm. Signatures are used to authenticate and safeguard requests.

Sender Token

Payment token created when buyers authorize purchase on their own behalf, often with a Pay Now button.

string-to-sign

Prior to calculating the HMAC signature, you first assemble the components for the signature in a sorted order, and then URL encode them. The pre-encrypted string is the string-to-sign.

website owner

A developer who uses Amazon Flexible Payments Service.

Document History

This documentation is associated with the 2010-08-28 version of the Amazon FPS Advanced Quick Start. This guide was last updated on 10-December-2012.

The following table describes the important changes since the last release of this guide.

Change	Description	Date
Enhancement	Added minor changes and typographical fixes applied from In this release a maintenance edit.	In this release
Feature Deprecation	Amazon FPS has removed support for signature verification using signature version 1. If your application is using this feature, you must convert to a server-side call with Verifying the ReturnURL and IPN Notifications.	2011-02-10
Feature Deprecation	Amazon FPS has removed support for client-side signature verification using PKI. If your application is using this feature, you must convert to a server-side call with Verifying the ReturnURL and IPN Notifications.	2011-02-10
Maintenance Update	Added an expanded breadcrumb to the HTML version, and reduced the front matter content. In addition several minor fixes and corrections have been applied.	2010-11-01
Enhancement	The RecipientVerificationStatus data type (used by the GetRecipientVerificationStatus action) has been enhanced to enable you to determine whether a verified customer account is unlimited in the amount of money it can receive. For more information, see “GetRecipientVerificationStatus.”	2010-11-01
Feature Extension	Amazon FPS has extended support for signature verification using signature version 1 to 10 February 2011. If your application is using this feature, you must convert to a server-side call with Verifying the ReturnURL and IPN Notifications.	2010-11-01
Feature Extension	Amazon FPS has extended support for client-side signature verification using PKI until 10 February 2011. If your application is using this feature, you must convert to a server-side call with Verifying the ReturnURL and IPN Notifications,	2010-09-14
Enhancement	Changed the values for soft descriptors from "AMZ*" and "AMZN PMNTS" to "ASI*" and "Amazon Payments", respectively. Also, applied minor changes and typographical fixes applied from a maintenance edit.	2010-06-11
Enhancement	Examples of the email messages sent by Amazon Payments which are relevant to Amazon FPS are now included as part of this guide. Please see “Email Notification Templates.”	2010-01-29
New Feature	GetRecipientVerificationStatus enables you to test that the intended recipient has a verified Amazon Payments	2010-1-15

Document History

Change	Description	Date
	account before you present the payment option for that seller or recipient on your web site. For more information, see “GetRecipientVerificationStatus.”	
New Feature	Support for signature version 2, which completely replaced signature version 1 on 10 February, 2011. The enhanced security features include: • a more secure way of calculating signatures for inbound requests and outbound notifications. For more information, see “Working with Signatures.” • support for SHA256 signing algorithm • the new VerifySignature FPS Action for server-side testing of return URL responses and IPN notifications. For more information, see “VerifySignature.” • support for PKI based authentication for client-side testing of return URL responses and IPN notification. Note This feature is deprecated as of 2010-09-14.	2009-11-03
Enhancement	The Access Keys page has been renamed the Security Credentials page, located at http://aws.amazon.com/security-credentials.You must be logged in to view this page.	2009-09-09
Feature Deprecation	Aggregate Payments, the AWS Developer Resource, and AWS Support Centers have been deprecated.	2013-08-08
Editorial Update	Added language to clarify that the Amazon Payments service has been designed and developed for use within a web browser only. Our service cannot be used within a native application (including, without limitation, iOS, Android, RIM and Windows operating systems).	2013-10-18
Correction	Fixed a typo in the description of the unverifiedEmailAddress_Sender error response.	2013-12-06